COMMODORE MAGAZINE

VOL. 3
ISSUE 2

MAY, 1983

VIC 20 VOTED COMPUTER OF THE YEAR

yes

Registered by Australia Post
Publication No. NBP4288 -

REGISTERED FOR POSTING AS A PUBLICATION: CATEGORY B

The objective of this magazine is to disseminate information to all users of Commodore computer products. This magazine contains a variety of information collected from other Commodore publications, and generated locally. Contributions from all Commodore User Groups, and individual users are encouraged.

Advertising space is restricted and allocated on a first-come, first-served basis. Advertising rates can be supplied on request.

All copy and advertising should be addressed to:

The Editor
COMMODORE MAGAZINE
COPUTER REFERENCE GUIDE
284 VICTORIA AVENUE
CHATSWOOD 2067 NSW
AUSTRALIA

ISSUE No.	COPY/ADV DEADLINE	FOR PUBLICATION IN
1	February 18th	March
2	APRIL 11	MAY
3	JUNE 3	JULY
4	JULY 29	AUGUST
5	SEPTEMBER 23	OCTOBER
6	OCTOBER 14	NOVEMBER

(Publication dates and advertising deadlines are currently under review.)

SUBSCRIPTIONS

	Annual Subscription	Single Copy
Postage paid within Australia	$A30.00	$A5.00
Overseas Postage Paid	$A38.00	$A6.00

Subscriptions to COMMODORE MAGAINE can be obtained from your authorized Commodore Dealer, or from:

COMMODORE MAGAZINE
COMPUTER REFERENCE GUIDE
284 VICTORIA AVENUE
CHATSWOOD 2066 NSW
AUSTRALIA

Publisher:–
Computer Reference Guide
284 Victoria Avenue
Chatswood
2067 NSW
Phone (02) 419 3277

Production & Typesetting:–
Computer Reference Guide
284 Victoria Avenue
Chatswood
2067 NSW
Phone (02) 4193277

Printer:–
M.A.P.S. Litho Pty Ltd
Cnr Gibbes and Smith Sts
Chatswood
2067 NSW
Phone (02) 406 6433

Vol 1 1981
Vol 2 1982
Vol 3 1983

COMMENT

Hard on the heels of the announcement of the production of the 1,000,000th VIC–20 comes the news that the VIC–20 has been voted "Home Computer of the Year" by the readership of 7 international personal computer magazines.

To VIC–20 owners this result will not be surprising, nevertheless it is gratifying to know that the excellence of Commodore products is internationally recognised.

Included in this edition are descriptions of new products announced by Commodore at the Consumer Electronics Show at Las Vegas. Once more Commodore demonstrates the innovation which keeps it at the forefront of microcomputing.

If you have any contributions in the form of program listings or articles on aspects and applications of your Commodore computer, which you think may be of interest to other Commodore users, please do not hesitate to send them in to us. Try to ensure that program listings are as accurate as possible.

Also don't forget to keep those game scores rolling in for our High Scores section. So far we have had only a few replies, but we know you're out there! So get those scores in – and remember you're on your honour to be accurate.

Table of contents

How to make the best home computer in the world even better.

Peripherals to turn a powerful computer into a super-computer for the professional.

With the Commodore VIC 20 Computer, you have the finest home computer money can buy. And the more you use it, the more you will ask it to do.

Pretty soon, you'll want to extend VIC's vast potential to the full; and there is a wide range of VIC peripherals to help you do it.

Disk drives, disk-based software, a printer, cassette unit, joysticks, paddles — with these, VIC computing becomes total computing: giving you true professional power and capability.

VIC PRINTER

The VIC Printer, like all VIC peripherals, offers a very high specification at a very competitive price.

It will print programs, letters, business data, graphic displays and so on.

Its main features include: 80 characters per line • Tractor feed dot matrix • 30 characters per second print speed • Full alphanumerics and graphic printing • Double-size character capability.

VIC FLOPPY DISK UNIT

The VIC single-drive Disk Unit provides a fast, accurate and efficient means of storing and retrieving data and programs.

Together with the Printer, it transforms the VIC 20 into the ideal system for the small businessman or serious computer programmer.

Features include:
174,848 bytes capacity
• Uses soft-sectored standard 5¼" single density floppy disks
• Direct interface to VIC
• Direct compatibility with Printer
• Intelligent system independent of VIC.

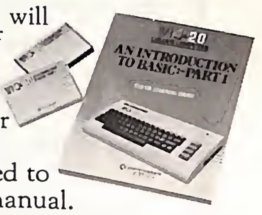
EXPANSION MEMORY CARTRIDGES

Special plug-in cartridges are available to expand VIC's memory. 3K, 8K and 16K RAM packs plug directly into the computer.

A Memory Expansion Board is also available to develop VIC's capabilities to the maximum.

1. Introduction to Basic

This package contains materials that will enable you to learn the fundamentals of programming in the BASIC computer language on your VIC 20 computer. It assumes no prior knowledge of computer programming and includes two cassettes containing 17 programs specially designed to accompany a comprehensive 152 page manual.



2. Progamming Aids

To aid programmers to write their own programs, Super Expander, Machine Code Monitor and Programmers Aid cartridges are available.

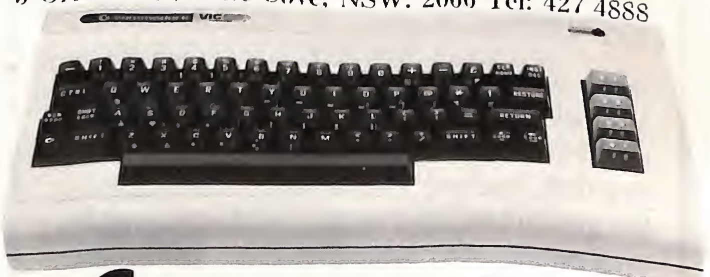
3. Games

The VIC 20 has the largest selection of cartridge and cassette-based games available on any personal computer. VIC 20 games offer a real challenge and take a great deal of skill, time and mental agility to win.

4. Joysticks and Paddles

These accessories make playing games on the VIC even more enjoyable. The joystick has even more practical uses and can be used with high resolution graphics to draw pictures or help with graph plotting.

For full details of VIC 20, its peripherals and software, and a list of your local dealers, contact:
The Commodore Information Centre,
5 Orion Rd., Lane Cove, NSW. 2066 Tel: 427 4888



commodore
VIC 20
A real computer for the price of a toy.

MLVL 1613

Commodore Announces One Millionth VIC 20 Computer

Featured on the middle of the Commodore booth at the winter Consumer Electronics Show was the one millionth VIC 20 computer, thus confirming the prediction made at the Commodore press conference one year earlier that "Commodore would sell more computers in 1982 than the entire industry did the previous year."

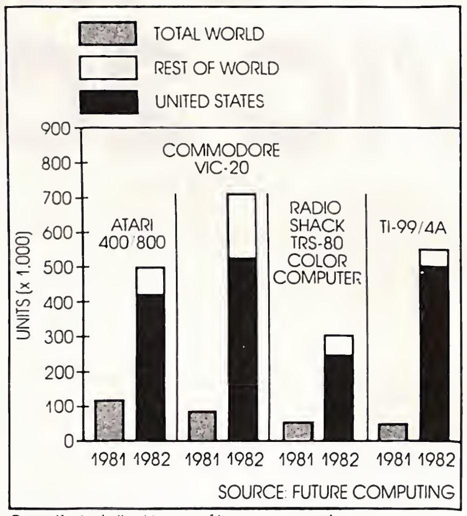
VIC 20 Voted Computer of the Year

An International competition run by 7 personal computer magazines has voted the Commodore VIC 20 "Home Computer of the Year." The magazines sponsoring the competition were:—"Databus" (Holland), "Microsystems" (France), "Bit" (Italy), "Practical Computing" (Britain), "Chip" (Spain), "Personal Computing" (U.S.A.), and "Chip" (Germany).

Computers had to meet basic criteria—fully developed machines, not pre-production prototypes; available to the marketplace with software and service back-up. Second and third places were awarded to the Sinclair ZX 81 and Spectrum, with the Atari 400 and Tandy color machines coming in as runners up.

Future Computing Inc Analyzes The Personal Computer Market

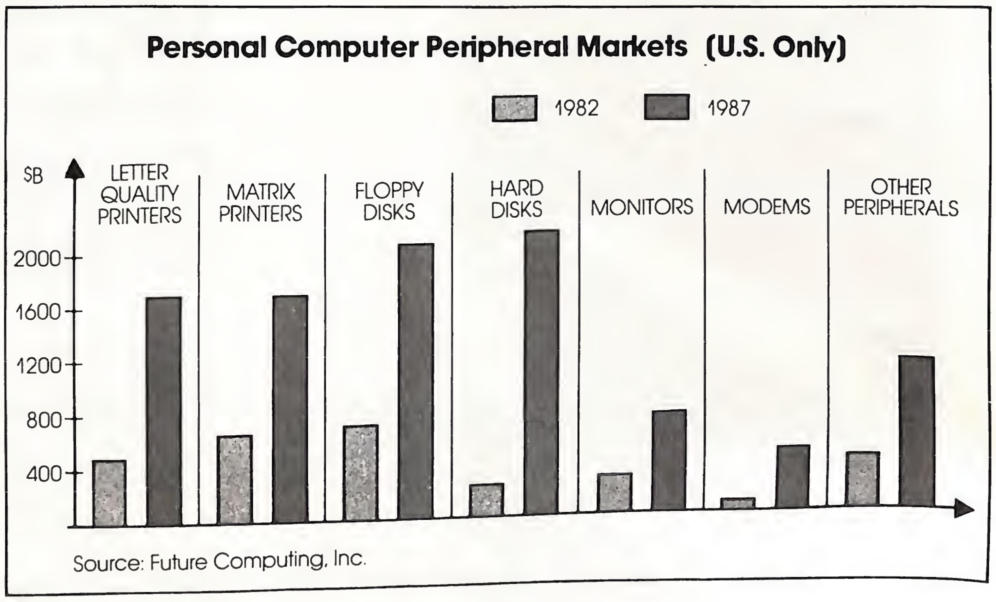
This record milestone for the industry and Commodore was made at the same time as the Dallas based market research firm FUTURE COMPUTING released their latest report on the personal computer market. The report included the comment that "one year ago the Apple II was the leading personal computer with the largest installed base, the largest shipment rate and the largest revenue generator of any single personal computer." The report also stated that "one year ago the Atari 400/800 was the clear leader in the home computer segment." The report then went on to comment that "Now one year later the changes have been phenomenal and that the monthly shipment rate of the largest home computer manufacturer (Commodore) has now outdistanced the Apple II by a factor of three or more." One result they forecast is that a flood of software will become available for the leading home computers in 1983. Judging by the quantity and variety of titles recently released by Commodore we should be leading the way to make this particular prediction come true.



Growth. Installed base of home computers —those selling for under $500

The Growth Opportunity in Peripherals

According to FUTURE COMPUTING's research report "the market for computer peripherals represents a major growth opportunity." In 1982 they valued it at $2.5 billion in the U.S.A. at retail price level. This they estimated was presently two-thirds of the total hardware market and will grow to over $10 billion by 1987. The three largest sectors were given as letter quality printers, dot matrix printers and floppy discs. While the new emerging sectors were given as hard discs, monitors and modems—all three of these are market sectors that Commodore has entered with competitive products over recent months. Commodore is already established in the first three major categories. So with the largest installed base of any computer manufacturer worldwide the company should be in a strong position to capitalize on these profitable markets in future years.



A Message From Our President and Founder

Our company has had outstanding success since we started 25 years ago. We are an exciting company in an exciting industry. However the rapid growth we have planned for and are experiencing means that I can no longer meet and get to know all of you personally to tell you of the constant principles that we have learned to rely upon and guide us through the many changes necessary to achieve and continue our rapid growth. These principles lie at the heart of Commodore's success. Learn them and learn them well and you will be successful as a member of the Commodore family. Practice them, discuss them with your colleagues and managers and we will continue as a leader and innovator.

[illegible]

JACK TRAMIEL
President and Founder
Commodore International

The Commodore Philosophy

THE COMMODORE PHILOSOPHY is like our company religion. Commodore is a "family" of employees so when we talk about ourselves we talk about "we" not "I." We should all share the same goal . . . to produce the best products for the best prices, using the newest available technology.

BUSINESS PHILOSOPHY . . . Business to us is not a sport—it's war. We are not here to play the sport but to win the battles. We will win because we work harder, work smarter and we serve our customers better. Every competitor, from the smallest to the biggest, is an enemy and we are dedicated to winning.

INTERNATIONALISM . . . We don't believe in nationalism. We believe in internationalism. Commodore is an international company.

DECISION MAKING . . . Progress at Commodore is maintained by middle management making the decisions that drive us forward. Every time we make a decision we have to look to the left and right to see if we are helping each other . . . not just thinking of number one (ourselves) because we as a group of over 3000 employees are much stronger than one.

CUSTOMERS . . . Our marketing strategy is clear. We produce for the masses—not the classes. Quality and service is our commitment. Our customers are mature and intelligent. We must give them the best because they will know if we don't.

TECHNOLOGY . . . Commodore is driven by technology. We don't only introduce the products the customer wants . . . we introduce products the customer didn't even know were available.

PROFESSIONALISM . . . A good professional needs to practice every day . . . not only to have one good "concert" a year. Don't just react—think first! Treat every situation as if you were the one being affected by your actions.

INVOLVEMENT . . . Delegating is important but involvement is more important. We all have to involve ourselves in the problems as well as the successes.

MUTUAL IMPROVEMENT . . . Never settle for doing things the way they were done in the past. Always find new ways to do things better, cheaper and more efficiently. Constructive criticism of each other is essential. We should not tell ourselves and each other only what we have done well. Every morning we should tell ourselves what we didn't do and what we could have done better.

ADVICE . . . No one is too important to accept advice. No one is too unimportant to give it.

COMMUNICATION . . . If you have information that others need . . . share it! Communication is a key strength of our company. There are no secrets inside the company but outside we keep our business confidential.

SUPPLIERS . . . Never buy a product if you don't know what it costs to make it. Never buy below the supplier's cost so the supplier is driven out of business. Give him a decent profit . . . but never more than our company makes. Develop long term relationships with our suppliers and always negotiate in good faith.

FINANCIAL RESPONSIBILITY . . . Treat every penny as your own.

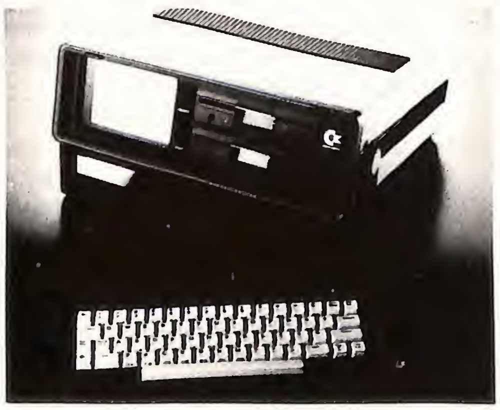
EMPLOYEE RELATIONS . . . Every employee should and will know the goals we are trying to achieve as a company. There is always a reward for the best but there is also a reward for the one that tries the hardest . . . because trying hard is important. Neither discrimination nor prejudice is tolerated at Commodore. Your individual problems are the problems of the company . . . we're here to help in happiness and sorrow. That's what a family is all about.

THE FUTURE . . . We're always looking at the future because we're helping to create that future . . . but the work is always done in the present.

Commodore Stuns Press At Winter Consumer Electronics Show

—Infoworld Magazine

This was the headline that Infoworld Magazine used to report on a Commodore press conference during the Consumer Electronics Show. Their opening paragraphs read as follows:
LAS VEGAS, NV. "A dual processor portable computer at a low price, the announcement of the Commodore 128 (P500), one million VIC's sold, the possible takeover of Zilog and scads of new products were among the topics at a Commodore hosted press conference at the winter C.E.S. Nearly 100 reporters packed a small conference room to hear Commodore spokesmen declare that the company is now number one in both home and personal computer sales. President Jack Tramiel boldly predicted that his company will be number one in home, personal and business computer sales by this time next year."

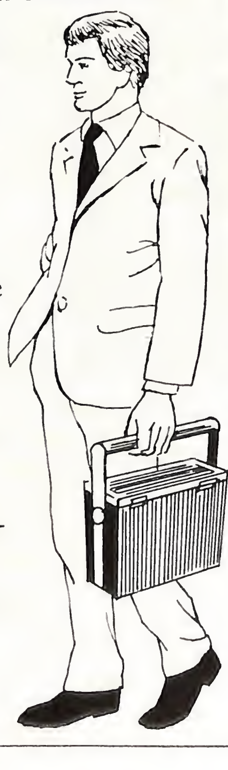


New Products Shown

Here we take a brief look at the new products that made their debut at C.E.S. and helped make Commodore the hit of the show.

The Executive 64-A Portable Version of the Commodore 64

The new product hit of the show was a briefcase style portable computer based on the popular Commodore 64. The portable computer shown at the booth had 64K of memory, a 5 inch color monitor and two standard size floppy disc drives (340K capacity) all built and packaged in an extremely attractive briefcase style housing complete with keyboard. The unit was both smaller and lighter than the leading Osborne computer on the market today. The computer was put together by a design team in Japan cooperating with their American counterparts in Commodore.



Compatibility with the rapidly expanding range of software for the Commodore 64 was evidenced each morning when Commodore staff took the same discs they used to demonstrate the standard Commodore 64's on the booth and inserted them in the new portable model. As well as the standard Commodore MOS 6510 microprocessor the 64 accommodates a Z80 processor and CP/M operating system.

While the final price of the Executive 64 has yet to be announced Mr. Gould, Commodore's chairman, commented that "the price of this series of machines, like all Commodore computers will be substantially below any comparable product now on the market." Certain configurations could well be on the market for under $1,000 while a full configuration with two disc drives and color monitor should retail for under $1,500. Initial shipments are being targeted for spring/summer.

Color Monitor for Under $300

At the CES show Commodore Business Machines U.S.A. introduced a low priced color monitor especially designed for its line of home, school and business computers.

The new 13 inch monitor—designated the CBM 1701—will retail for $299.95 and was especially designed for use with the COMMODORE 64 and VIC 20 computers. The monitor accepts a standard 75 ohm composite video signal or a "Commodore" video signal with separate provisions for luminance and chrominance signal input as well as audio input.

The monitor was developed in conjunction with a major television manufacturer and includes special circuitry which greatly enhances the picture resolution. Commodore has applied for a patent on the design.

Up until now the main choice for a user has been to either buy an expensive monitor or to keep connecting and disconnecting the family T.V. set every time.

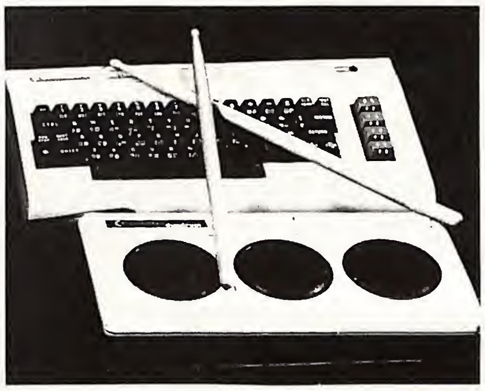
From a business standpoint Commodore has diversified into color monitors as they will become a key component in future color computer systems. So now retailers and customers will be able to buy their full system from Commodore just like stereo component systems are sold. Deliveries of the color monitors are scheduled for early 1983.

A Low Cost Printer/Plotter

Also at the CES Show Commodore unveiled a new printer/plotter to retail at $199.95. This computer accessory uses 4½ inch roll paper and prints in four colors or combinations of colors to achieve multi colored graphs, charts and other types of illustrations which are enhanced by the use of color. High resolution illustrations are achieved by the printer/plotter's ability to "step" 480 dots horizontally and up to 999 steps vertically. The unit can also print out letters and numbers as required.

Four separate ball point ink pens provide a clean, high quality color image and the 5 inch wide carriage accommodates standard roll paper. The VIC 1520 printer/plotter rounds out the existing line of low priced main accessories for the VIC and COMMODORE 64 computers. First deliveries are scheduled for early 1983.

Digidrum Adds New Dimension



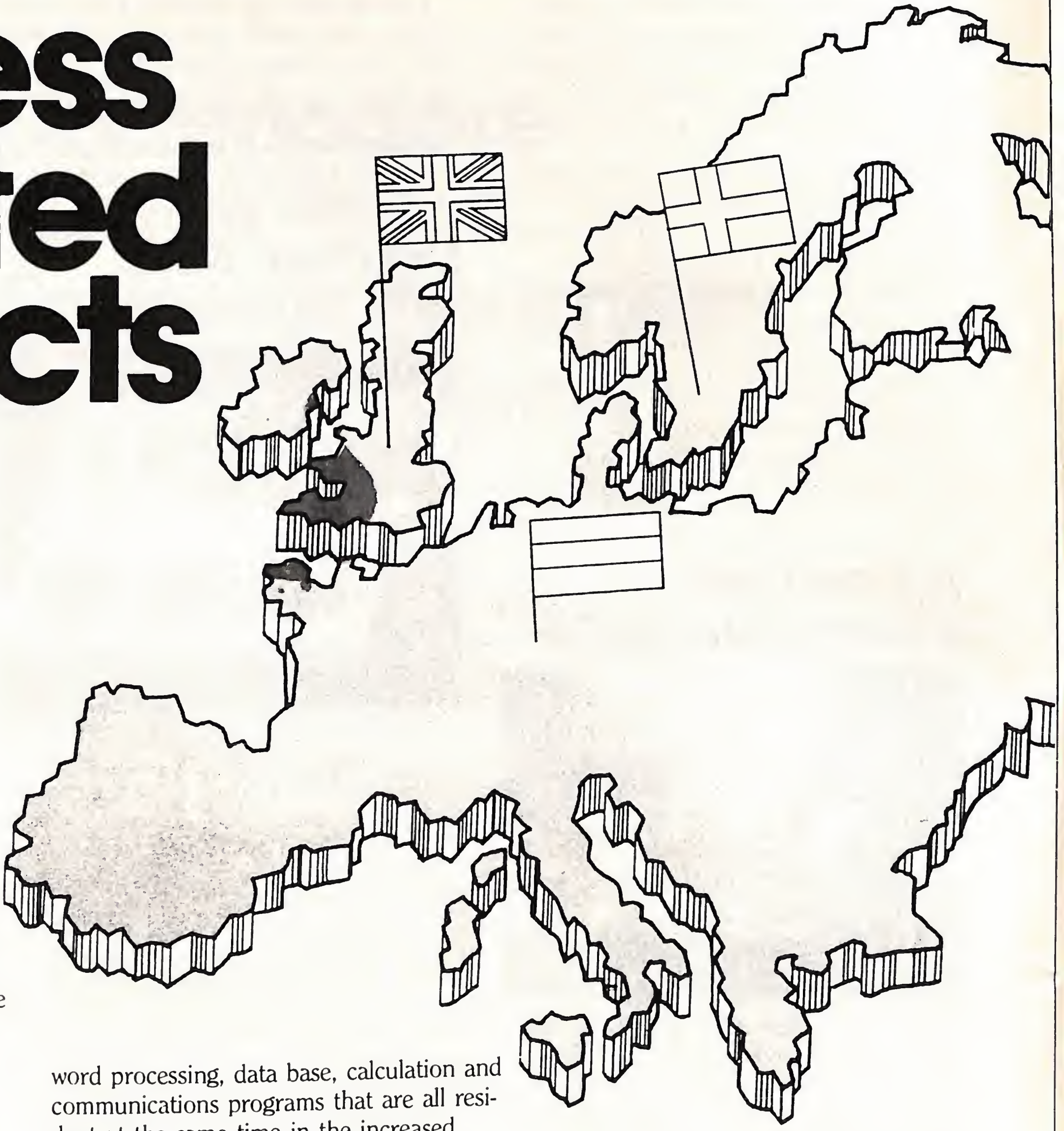
Imagine an electronic drumset controlled by a computer, with volume control and sound generated through a TV set, monitor or stereo system . . . but priced as low as a video game cartridge! At the CES Show Commodore demonstrated a new low priced three-pad electronic drumset called DIGI-DRUM™ which will attach to the COMMODORE 64 and VIC computers. This new peripheral will plug into the computer's expansion port and comes complete with special software which lets the user simulate a snare drum, base drum and "high hat" cymbal, with startling realism.

The combination of computer and DIGI-DRUM visually displays 3 animated drums on the TV screen, which perform with each drumstroke. Other screen simulations can be programmed. The three drums can be combined in an infinite variety to produce high quality rhythm effects suitable for entertainment, learning or sound effects. DIGI-DRUMmers can use their computers to create and save drum routines and play them back through their stereo system or TV speaker. First deliveries are planned for Spring/Summer 1983.

A Keyboard Synthesizer for Under $100

Another new product demonstrated at the CES show was a 12 voice, 37 full size key organ keyboard synthesizer designed to work with both the VIC and 64 computers. Accompanying the keyboard is an interface card that plugs into the cartridge port of the computer. A demonstration tape for the unit allows the user to play and select from a few instrument parameters. In addition there is to be a cartridge slot in the interface itself into which special music cartridges can be inserted. The first of these will allow the user to select from unlimited preset instruments. Users can also record what they play in real time and then play back and accompany their recordings. The computer screen can be used to highlight various aspects including a graphic display of the synthesizer with slider controls. While the price on this unit is not yet announced it is anticipated to be available by the summer at under $100. This together with the DIGIDRUM brings a new world of spectacular music within the reach of would be personal computer owners and opens up the market for electronic music to Commodore.

Business Oriented Products Lead Off 1983



While Commodore made dramatic gains to world leadership in home computers last year much backroom work was going on in its business oriented products to support the traditional dealer base and get ready for strengthened activity in that area during 1983.

Upgrated CBM Computer Becomes Workhorse of the Industry

For several years the CBM 8032 desktop computer has been the backbone of our business sales. Especially so in Europe where it is the brand leader. During 1982 a 64K add on memory board was brought into the range. This has enabled some very powerful upgrades to be made to the large existing base of software for this machine. These upgrades include: a version of Visicalc running with 96K of memory to give an exceptionally large worksheet and data base, a version of the popular Wordcraft word processor package with built in communications facility to other computers, plus the much acclaimed Silicon Office from the U.K. Silicon Office is a fully integrated package of word processing, data base, calculation and communications programs that are all resident at the same time in the increased memory facility of the computer. Indeed there has even been a new electronic spreadsheet package developed by our distributor in Sweden which has several advantages over the enhanced Visicalc.

As a result of this Commodore companies in Europe have marketed the increased 96K version of the computer as a separate machine in its own right—the CBM 8096 Business computer.

Commodore 128K Color Computer

Initial shipments of the new P500 series computer, named the Commodore 128 in the U.S.A. are to begin during the second quarter of 1983. Priced at $795 in the U.S.A., this computer is expected to achieve the same rapid acceptance as the Commodore 64 has done. The 128K computer offers all the popular features of the 64 including color, music synthesis, sprite graphics and a CP/M option. Further enhancements make it particularly useful for business and scientific needs. These additions include both IEEE–488 and RS232 interfaces built in. The former makes it fully compatible with the full range of Commodore CBM peripherals already on the market. These include hard and floppy disk units up to 7.5 megabyte capacity, printers and a wide range of scientific instruments such as Hewlett–Packard devices. Also included are 10 function keys in an extremely attractive and functional styling. This computer is being sold on a restricted distribution basis via professional Commodore dealers.

An 80 column monochrome version of this model, based on the "B" series, will also be marketed.

Commodore "B" & "Bx" Series Business Computers

Initial shipments of the B700 series computers will begin in mid 1983 to boost Commodore's presence in the business sector. Acclaimed worldwide for the extremely attractive and practical styling which includes a detachable keyboard and a screen with both tilt and swivel facilities, the B700 series has just as impressive technical features as well as outstanding price.

The B700 series comes with either 128 or 256K built in user memory and can potentially be expanded up to a massive 896K total internal memory. This series has a full 80 column green monitor display and like the "P" series is fully compatible with the wide range of existing CBM peripherals as well as having its own optional built in floppy disc drives.

In addition to the standard Commodore 6509 microprocessor the computer has also been designed to take a second co-processor. Both a 16 bit 8088 system and a Z80 system will be offered by Commodore as extras—the former will be offered as a built in standard on the "BX700" series.

The wealth of software for the CBM 8000 series is currently being upgraded and enhanced to take advantage of the additional capabilities of this new series. The "B" & "BX700" series computers will be distributed through Commodore professional business dealers and will spearhead our attack on the business markets in the second half of 1983.

FUND RAISING OPPORTUNITY

We have a self liquidating scheme that could earn approx $9,000 over a period of twelve months for a club or charity etc. that is recognised by its local community.

Operational boundaries are:-

★ A community of at least 6,000 homes (10,000 optimum)
★ A local industry or retail base that could support 120 x $50 donations (in return for a small promotional service)
★ At least ten reliable, hardworking and positive thinking workers.

Benefits on top of Funds raised:–

★ Where possible local industry and labour are used
★ Has excellent public relations possibilities
★ Expandable and can be continued beyond the first twelve months.
★ Virtually self liquidating

Note: This is not a get rich quick idea – it does require time and a lot of effort.

Ideal for Scout groups, Churches, Charitable organisations.

MERVYN BEAMISH GRAPHICS PTY LTD
Suites 102-105, 82 Alexander Street, Crows Nest 2065 NSW
(02) 439 1827

PHOTOTYPE SETTING ART & DESIGN SERVICES
Let us put you on the mailing list for our irregular but informative Newsletter.
WE CAN PHOTOTYPE SET DIRECTLY OFF MOST COMMODORE GENERATED DISKETTES AND CASSETTES.

From Typewriter Repair to World Leader in Electronics

The story of Commodore begins 25 years ago and is closely associated with Jack Tramiel, the founder of our company and architect of what it has become today.

Jack's first association with our industry was in his army days at Fort Dix where he became involved with the repair of typewriters. This was back in the days of mechanical machines and long before word-processors were thought of. However, the experience in the applications and the technology gave a useful base for the future and it was this business that Jack went into after leaving the army. Ever one for hard work, Jack Tramiel also drove a New York taxi in these early struggling years to establish himself in civilian life. It was the typewriter experience that led to the start of Commodore a few years later when Jack moved his family to Toronto, Canada, and started his own typewriter repair business. The year was 1958 and it was the beginning of the Commodore International of today.

The First Ten Years

During those early start-up years some 25 years ago, Commodore progressed from typewriter repairs to typewriter selling and eventually to typewriter manufacturing with the acquisition of a factory in Berlin, Germany. This was Commodore's first major foothold in the European market that would become very important to us in future years. Responsive to our customer needs for office equipment Commodore also moved into mechanical adding machines imported from Japan. This gave us our first experience of the Far East and the industrial potential that was to come from there. Indeed while many Western companies are now trying to understand the "Japanese phenomenon" it is comforting to know that we have had our own Japanese company for more than a decade. Commodore Japan has been, and still is, making valuable contributions to the growth of our international organization.

The Electronic Calculator and the Second Decade

In 1965 Jack Tramiel met the Canadian financier Irving Gould who was to become Commodore's chairman and the two of them the top management team that has stayed together for over 15 years to build the Commodore we know today.

It was shortly after this that Jack got his first look at what was then a large desktop ELECTRONIC calculator. Jack Tramiel realized that it would one day spell the doom of mechanical machines and open up new business potential. With the Commodore philosophy that "if we are not our own competition then someone else will be" we went fully into the electronic desktop calculator market. So once again when the first electronic "pocket" calculator appeared a few years later by combining a Texas Instrument chip with a Bowmar l.e.d. display it was brought to market by Commodore as the C108 "hand held" calculator. It was another example of what has become a long history of Commodore "industry Firsts" in marketing value, innovation and performance in new products.

Commodore sales grew dramatically during this period from 1969 to 1974 with the launch of this first "under $200 electronic calculator." It is interesting to note that this was sold at much the same price, through similar distribution channels and to similar customers as does our "under $200 VIC computer" today. During this period we developed a strong distribution system in Europe that has remained a strength of ours ever since.

Our line of calculators grew from simple 4 function machines, to memory machines, scientific machines and to keyboard programmable models. So the stage was set for the next logical step—a full computer.

However up to this time Commodore had been largely dependent on third parties for the supply of the crucial chips and displays that went into the products we were making. So in 1974/5 we suffered badly with a rapid decline in the price of calculators and components as some of the makers of components tried to enter the market for end products along with Commodore and dumped their inventories onto the market. These hard times lead Commodore to do some key strategic thinking.

The Drive for Vertical Integration

Commodore was convinced of the long term potential for semiconductor based office and consumer electronic products. We were also convinced that technology would be a key driving element. So a major decision was made to become "vertically integrated." This is the name in our industry for producing both the key electronic chips and the final user products they go into.

With hindsight this may seem to have been an obvious decision. It was not so readily apparent at the time when both the calculator and semiconductor markets were suffering badly and every bit of finance was needed to keep the day to day operations running. Indeed it was at this time that our chairman personally guaranteed a $3 million loan to the company in order to make the critical investments for our future growth.

The Purchase of M.O.S. Technology

In 1976 we made our first major purchase of MOS Technology—a struggling manufacturer of calculator and other semiconductor chips based in Norristown, Pennsylvania. This purchase for under $1 million proved to be an extremely wise decision for the marriage of their technology with Commodore's business, marketing and mass production skills proved to be a formidable combination. This acquisition was followed in the next eighteen months by two further key investments. These were the purchase of Frontier, a Los Angeles based manufacturer of C-MOS chips which is a complimentary technology to the N and P-MOS chips of MOS Technology. The other major acquisition was that of M.D.S.I., a Dallas based manufacturer of liquid crystal displays. This was another key technology that was to become the basis of nearly all calculator displays and holds much future potential including use in computers. The result of all this was that Commodore had in house expertise and production in more key technologies than most electronic companies several times its then size.

Our Entry Into the World of Computers

During this time we were deciding on what our next programmable scientific calculator should look like. At the same time we were reviewing the potential for KIM 1, a microcomputer board from the recently acquired MOS Technology that had achieved some success in specialized market areas. The bold decision was made to develop our own fully fledged microcomputer. A small team got feverishly to work and in record time the PET (Personal Electronic Transactor) was developed and launched to the world in 1977 at the Hanover Show in Europe and the Consumer Electronics Show in the U.S.A. It was this machine that together with the TRS 80 and the Apple was to give birth to the personal computer marketplace of today. The PET computer used our own MOS Technology 6500 microprocessor that has become the heart of many other computers since including Apple and Atari. While the PET's technical features were exceptional so were some of the user friendly features that are a hallmark of Commodore Computers. These included the built-in screen, built-in basic language, cassette deck, graphics characters, intelligent peripherals and easy screen editing. Many of these features are still found on our latest models. Indeed a program written in Basic on the original PET will run unmodified on our new Commodore 64—an amazing degree of compatability considering the progress made in product value and features.

The Rise to World Leader

Commodore's entry into the computer market once again sparked off a period of rapid growth that is still in its infancy today. The Commodore PET was marketed worldwide but a strategic effort took place in the complex European market where we very rapidly established the leadership position via a strong dealer network for the PET and CBM Business computer.

Commodore is not a company to sit back and congratulate itself so the effort stayed on to develop further new products. The next major computer product was the VIC 20—the first full featured color computer to be launched for under $300. With a mindful eye for "stopping the competition on the beaches" the VIC was first launched in Japan and it has proved to be yet another Commodore winner. The VIC was also the product used by Commodore to spearhead an attack on gaining a major share of the U.S.A. market starting in 1981. So successful has this been that by January 1983 over 1 million VIC 20's had been sold, the largest installed base of any computer ever, and Commodore had become the largest unit seller of computers in North America.

The story does not end there for in the latter months of 1982 we launched the COMMODORE 64 for $595 as a direct competitor to the world famous Apple II. By the end of the year and aided by the single biggest advertising launch in Commodore history it had already passed the Apple II in monthly unit sales. So Commodore has yet another winner to enter 1983 with to say nothing of the new "B" series business and "P" series personal computers being launched as this article is being written.

Commodore is About Creating History

If one studies the history of Commodore it is readily seen that we have consistently been a leader in recognizing change and leading our industry into it. But more than studying history Commodore is much more a company that creates the history and that's exciting for all of us in it.

New Technologies Shown

During the CES Show, Commodore took the opportunity to demonstrate prototypes of new technologies and products it is working on. These included a touch sensitive screen that can be used separately or overlayed directly on a computer screen and a "mouse" type controller currently being featured on some very expensive competitors products. No formal announcement about products or prices are being made at this time.

Commodore and Zilog Announce Agreement

Mr. Irving Gould, Chairman of the Board of Commodore International Limited has announced that Commodore has signed a definitive agreement with Zilog, Inc., an affiliate of Exxon Corporation.

According to Mr. Gould, "the agreement relates to an exchange of technology between Commodore and Zilog, with Commodore licensed by Zilog to manufacture Zilog's Z8000 16-bit microprocessor and related family of peripheral support circuits for use in Commodore microcomputer systems, while Commodore will supply Zilog with mask sets and manufacturing rights to selected custom circuits used in current and future Commodore microcomputer systems."

Mr. Gould went on to note that "Commodore's financial business commitment to Zilog is significant and involves a similar commitment by Zilog that could utilize a substantial portion of Zilog's production facilities devoted to the manufacture of large scale integrated circuits for exclusive use in Commodore microcomputer systems."

Mr. Gould concluded by stating that "this agreement with Zilog is intended to give Commodore a substantial increase in semiconductor manufacturing capacity toward meeting its large scale integrated circuit needs, while reducing the time and investment required if Commodore were to expand its own semiconductor manufacturing capacity."

"Notwithstanding this," Mr. Gould added, "our own semiconductor facilities capacity is in the process of being increased by 100% as part of a program to be completed over the next twelve months."

Gortek and the Microchips

Gortek Is Coming

Gortek is the main character in a new series of space adventure stories designed to teach BASIC programming. The first lesson consists of two cassette tapes containing 12 educational programs and a book, and is called *Gortek and the Microchips*™. The book includes imaginative full-color illustrations, large easy-to-read type, and is written so that it may be read by older children or used by younger children with parental assistance.

The unique combination of the storybook and computer lessons makes Gortek a fun experience for adults as well as children.

As the story goes, the planet Syntax is being invaded by the fearsome Zitrons. Gortek works furiously to teach the microchips to program the computer to repel the attack. The "Microchips Training Manual" teaches the child how to stop the Zitrons—by learning to program the computer!

Those who complete the lessons and defeat the Zitrons earn the right to wear the Gortek badge which comes in the package.

This innovative approach to computer education was developed by three English school teachers who wanted to make programming fun to learn.

START PROGRAMMING WITH
GORTEK AND THE MICROCHIPS
commodore
TO LOAD AND RUN A PROGRAM FOLLOW THESE INSTRUCTIONS:

Machine Language Disk Error Routine

For PET/CBM (Upgrade or 4.0 ROM's), VIC 20 or Commodore 64 computers.

by
Thomas Henry
Transonic Laboratories, 249 Norton Street
Mankato, MN 56001 507/387-1642

There's no worse feeling in this world than seeing the disk error LED light up and not being able to do anything about it! For example, the editor program, which is part of the Commodore Assembler Development Package, occasionally freezes up when asked to find a source file on disk that doesn't actually exist. (A name may have been mispelled, the wrong disk might be in the drive, etc.) When this happens, the error LED lights, the computer locks up, and anything in the computer is lost. The only way to recover from this is to power down and up again! Wouldn't it be much nicer if a disk error message was printed and control was returned to the user?

It's important to assume, in any type of programming, that the user may make such mistakes. To qualify as "user friendly," a computer shouldn't let such scenarios as that described above happen. To prevent things like that from on happening Commodore equipment, the following program was written. The program not only fixes the editor program mentioned above, but more importantly, can be added to any machine language program you may have. By changing the equates as shown, you will be able to get the program up and running on PET/CBM computers with Upgrade or 4.0 ROM's, the VIC 20 or the Commodore 64.

It's easy to provide disk error protection in BASIC programs; you just OPEN the error channel and read the status. But how do you do this in machine language? As it turns out it's quite easy to do, using just a handful of ROM routines already in your computer. Before looking at the assembler listing, let's define in very precise terms just what the program should do.

We want a routine that can be called by a JSR at any time. Typically, you would jump to this routine whenever a file has been opened. If the file opened successfully, control should return to the calling routine. If trouble was encountered (FILE NOT FOUND, WRITE PROTECT ON, FILE EXISTS, etc.), then the program should direct control to some sort of error routine. For example, the error routine might send you back to a warm start, or some other "graceful" exit function. So, when the disk error routine is summoned, if everything is okay no message is printed and an RTS returns control back to the calling program. If an error is detected, default devices are restored, the error message printed, and control is diverted to an error handling routine.

Having define what the program should do, we can examine the listing. Refer to it now. There are three sets of equates; pick the set that corresponds to your computer. Note that VIC 20 and Commodore 64 users share the same set of equates (except for one location). This is due to the fact that these two computers use the same jump table.

To start the routine off, the error channel is opened by sending a talk with secondary address of fifteen ($F = $0F OR $60). The disk drive will now spew out the error message, and the routine at label XFER picks off this message. Note that we accept the message whether it is good or bad news; later on we'll decide whether to print it or not.

The message is stored, character by character in a buffer. The tape buffer was used in this instance, but any other section of unused memory that's handy may be used.

A carriage return ($0D signifies that the end of the message has been found. The carriage return is stored, and then is followed by a zero byte. By doing this, we can call the routine PSTRNG later on, which will print out any message up to a zero byte. But more about that later.

At this point, an UNTALK is sent to the disk drive, and this clears it for further use. Next the program checks to see if the first two bytes in the message are ASCII zeros ($30). If they are, then no error has occurred, and we return to the calling routine immediately, with no message printed.

If, however, these zeros aren't found, then an error must have occurred. The devices are restored to their default values (keyboard and screen), and the message is printed out via PSTRNG. Next the stack is pulled twice, and this disables the return to the calling program. Finally a jump is made to the error routine mentioned above.

You can see, then, that it is actually quite easy to provide disk error detection in machine language. The routine, as presented in the listing, is a mere sixty-four bytes long, surely a small price to pay for "user friendliness"!

Even if you don't have an immediate use for this subroutine, it's recommended that you put it into your programming notebook for later use. The time will come sooner or later when you'll want to safeguard a program against disk failures. You can use this routine in programs of your own devising or add it to commercial programs. Either way, you'll welcome the protection from disk or computer lock-ups!

This routine has been implemented on a number of programs over the past year and has proved to be quite reliable. It has been used on both the CBM-8032 with a 4040 disk drive and the VIC 20 with a 1540 disk unit. In all cases, much hair pulling has been prevented! Here's hoping that your programs will never again suffer from disk drive errors!

```
LOC   CODE       LINE

0000              ;***************************************
0000              ;*                                     *
0000              ;*  GENERAL PURPOSE DISK ERROR ROUTINE *
0000              ;*       FOR COMMODORE COMPUTERS       *
0000              ;*                                     *
0000              ;*            THOMAS HENRY             *
0000              ;*       TRANSONIC LABORATORIES        *
0000              ;*          249 NORTON STREET          *
0000              ;*          MANKATO, MN 56001          *
0000              ;*                                     *
0000              ;***************************************
0000              ;
0000              ;
0000              ;*** EQUATES FOR UPGRADE ROM PET'S ***
0000              ;
0000              ; DEVICE = $D4           ;CURRENT DEVICE.
0000              ; BUFFER = $027A         ;TEMPORARY BUFFER FOR STRING.
0000              ; PSTRNG = $CA1C         ;PRINT STRING AIMED BY (A,Y).
0000              ; TKSA   = $F128         ;SECONDARY AFTER TALK.
0000              ; ACPTR  = $F18C         ;ACCEPT DATA FROM DEVICE.
0000              ; UNTALK = $F17F         ;UNTALK THE DEVICE.
0000              ; TALK   = $F0B6         ;COMMAND DEVICE TO TALK.
0000              ; RESTOR = $FFCC         ;RESTORE DEFAULT DEVICES.
0000              ;
0000              ;
0000              ;*** EQUATES FOR 4.0 ROM PET'S ***
0000              ;
0000              DEVICE = $D4           ;CURRENT DEVICE.
0000              BUFFER = $027A         ;TEMPORARY BUFFER FOR STRING.
0000              PSTRNG = $BB1D         ;PRINT STRING AIMED BY (A,Y).
0000              TKSA   = $F143         ;SECONDARY AFTER TALK.
0000              ACPTR  = $F1C0         ;ACCEPT DATA FROM DEVICE.
0000              UNTALK = $F1B6         ;UNTALK THE DEVICE.
0000              TALK   = $F0D2         ;COMMAND DEVICE TO TALK.
0000              RESTOR = $FFCC         ;RESTORE DEFAULT DEVICES.
0000              ;
0000              ;
0000              ;*** EQUATES FOR VIC-20 AND COMMODORE 64 ***
0000              ;
0000              ; DEVICE = $BA           ;CURRENT DEVICE.
0000              ; BUFFER = $033C         ;TEMPORARY BUFFER FOR STRING.
0000              ; PSTRNG = $CB1E         ;PRINT STRING AIMED BY (A,Y).
0000              ;                        ;CHANGE TO $AB1E FOR THE 64.
0000              ; TKSA   = $FF96         ;SECONDARY AFTER TALK.
0000              ; ACPTR  = $FFA5         ;ACCEPT DATA FROM DEVICE.
0000              ; UNTALK = $FFAB         ;UNTALK THE DEVICE.
0000              ; TALK   = $FFB4         ;COMMAND DEVICE TO TALK.
0000              ; RESTOR = $FFCC         ;RESTORE DEFAULT DEVICES.
0000              ;
0000              ;
0000                     * = $5000
5000              ;
5000  A9 08       DS     LDA #$08        ;GET READY TO READ
5002  85 D4              STA DEVICE      ;THE ERROR CHANNEL.
5004  20 D2 F0           JSR TALK        ;COMMAND DEVICE TO TALK.
5007  A9 6F              LDA #$6F        ;THIS IS THE ERROR CHANNEL.
5009  20 43 F1           JSR TKSA        ;SECONDARY ADDR. AFTER TALK.
500C  A0 00              LDY #$00
```

```
LINE# LOC   CODE         LINE

00060  500E  20 C0 F1    XFER   JSR ACPTR       ;GET BYTE FROM BUS.
00061  5011  99 7A 02           STA BUFFER,Y    ;SAVE IT IN BUFFER.
00062  5014  C8                 INY
00063  5015  C9 0D              CMP #$0D        ;LOOK FOR CARRIAGE RETURN.
00064  5017  D0 F5              BNE XFER        ;IF NOT, GET NEXT CHAR.
00065  5019  A9 00              LDA #$00        ;PUT ZERO BYTE FOR END.
00066  501B  99 7A 02           STA BUFFER,Y
00067  501E  20 B6 F1           JSR UNTALK      ;UNTALK THE CHANNEL.
00068  5021  A0 00              LDY #$00
00069  5023  A9 30              LDA #'0
00070  5025  D9 7A 02           CMP BUFFER,Y    ;CHECK FOR ASCII '0'.
00071  5028  D0 06              BNE BAD         ;IF NOT, MUST BE ERROR.
00072  502A  C8                 INY
00073  502B  D9 7A 02           CMP BUFFER,Y    ;CHECK FOR ANOTHER ASCII '0'.
00074  502E  F0 0F              BEQ GOOD        ;IF FOUND THEN NO ERROR.
00075  5030  20 CC FF    BAD    JSR RESTOR      ;WE FOUND AN ERROR SO,
00076  5033  A9 7A              LDA #<BUFFER    ;RESTORE DEFAULT DEVICES.
00077  5035  A0 02              LDY #>BUFFER
00078  5037  20 1D BB           JSR PSTRNG      ;PRINT THE ERROR MESSAGE.
00079  503A  68                 PLA             ;DON'T RETURN IF BAD, BUT
00080  503B  68                 PLA             ;JUMP TO USER DEFINED
00081  503C  4C 40 50           JMP ERROR       ;ERROR ROUTINE.
00082  503F  60          GOOD   RTS             ;OKAY TO RETURN AND CONTINUE.
00083  5040              ;
00084  5040              ;
00085  5040  00          ERROR  BRK
00086  5041              ;
00087  5041              ;THIS IS THE USER DEFINED ERROR ROUTINE.
00088  5041              ;FOR EXPERIMENTAL PURPOSES YOU MAY WANT
00089  5041              ;TO LEAVE THIS 'BRK' INSTRUCTION IN.
00090  5041              ;FOR YOUR FINAL PROGRAM, REPLACE IT
00091  5041              ;WITH AN ERROR ROUTINE THAT SUITS YOUR NEEDS.
00092  5041              ;FOR EXAMPLE, YOU MAY WANT THE ERROR ROUTINE
00093  5041              ;TO BE A JUMP TO A WARM START, ETC.
00094  5041                     .END

ERRORS = 00000

END OF ASSEMBLY
```

The Computer Becomes a Synthesizer

by Kent A. Multer

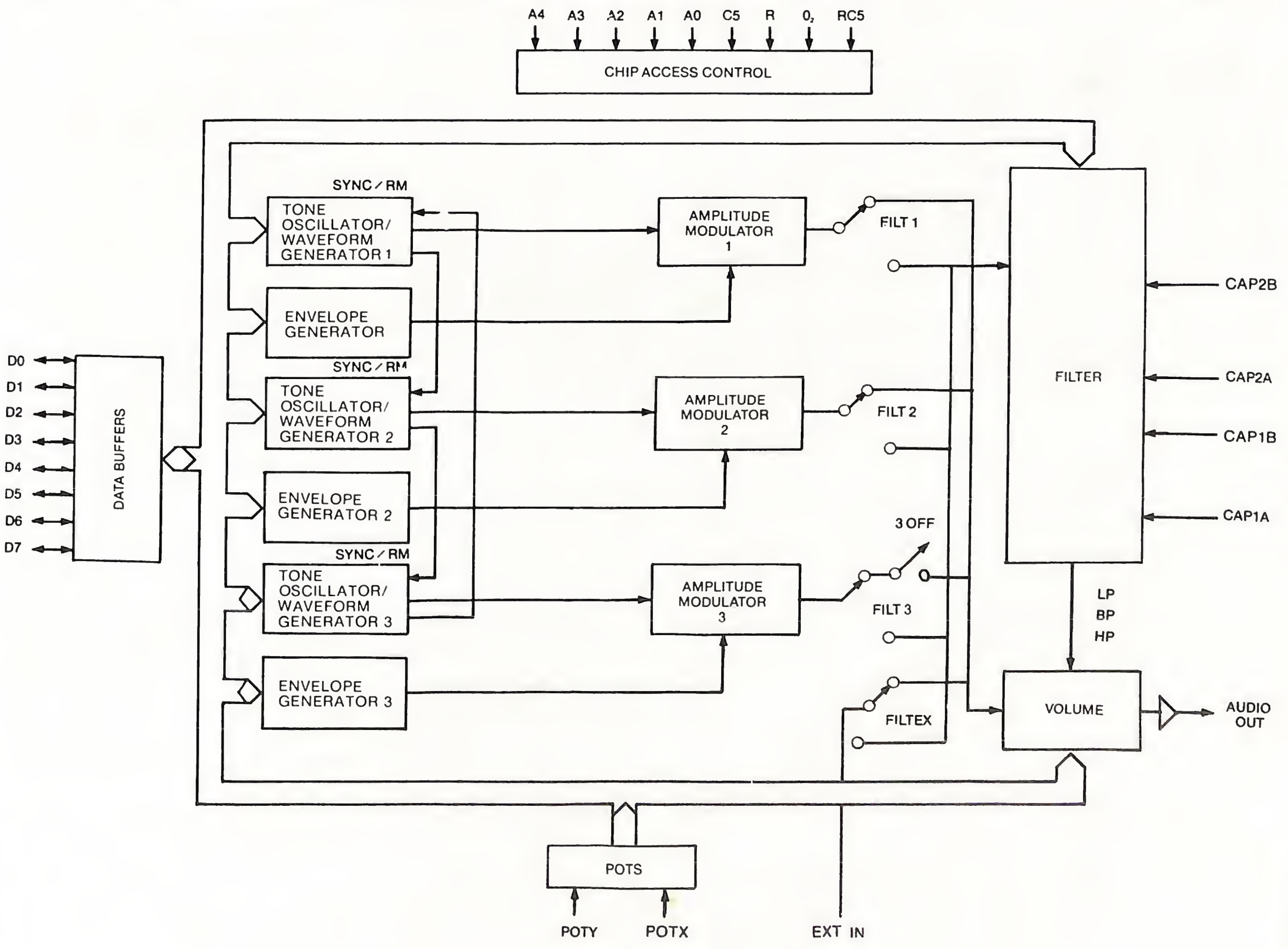
One of this year's most exciting new pieces of computer hardware is Commodore's model 6581 Sound Interface Device (SID). It's truly impressive: an entire music synthesizer on a single chip. SID is standard equipment in Commodore's newest computers, including the model 64, an advanced home computer with 64K of RAM. In this article I will describe SID's features, and give you some ideas on how it can be used.

Overview

Figure 1 shows a block diagram of the chip. As you can see, it has three voices, meaning it can produce three notes at once. Each voice consists of a tone generator, which produces the sound, and an envelope generator, which controls the volume. There are also some modulation effects, in which two tone generators combine to produce one complex sound.

The signals from the voices may be routed through a filter, which acts like a super tone control. This is the thing that makes SID so powerful, and so much more versatile

Figure **1** *Block diagram of the Commodore 6581 Sound Interface Device (SID).*



than the sound generators on other home computers.

Other features of SID include a master volume control, and an external audio input that allows you to run a signal from your electric guitar or other source through SID's filter. There are also two A/D converters on the chip, intended for connecting to pots. These are not electrically connected to anything else in SID, so you can use them for game controllers or whatever.

SID's functions are controlled by a number of 8-bit registers. In Commodore's machines, the chip is mapped into the memory address space, so you can write data into the registers with POKE statements. Each voice has seven registers for controlling its specific functions, and there are 8 more registers for controlling the filter, master volume, etc. Figure 2 illustrates the control registers.

Tone Generators
SID has three tone generators, each of which can generate four different waveforms: sawtooth, triangle, pulse, and noise. Sawtooth waves contain lots of harmonics, and are good for rich sounds like horns or strings. Triangle waves only have a little bit of harmonic energy, so they have a very mellow, flute-like sound.

Pulse waves can have a lot of different sounds, depending on the pulse width, which you can vary. When the pulse width is at or near 50%, you set a square wave, which is sort of hollow or clarinet-like. When the pulse width is close to zero (or to 100%; they both sound the same to the ear), the tone is very thin, more like an oboe. In between is an impressive variety of sounds, including saxophones, and sometimes human voices, if you set the filter right.

Figure **2** *SID control registers.*

Address ‡	DATA								REG
)	D7	D6	D5	D4	D3	D2	D1	D0	NAME
									VOICE 1
54272	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO
54273	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI
54274	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO
54275	—	—	—	—	PW_{11}	PW_{10}	PW_9	PW_8	PW HI
54276	NOISE				TEST	RING MOD	SYNC	GATE	CONTROL REG
54277	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK ⁄ DECAY
54278	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN ⁄ RELEASE
									VOICE 2
54279	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO
54280	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI
54281	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO
54282	—	—	—	—	PW_{11}	PW_{10}	PW_9	PW_8	PW HI
54283	NOISE				TEST	RING MOD	SYNC	GATE	CONTROL REG
54284	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK ⁄ DECAY
54285	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN ⁄ RELEASE
									VOICE 3
54286	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO
54287	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI
54288	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO
54289	—	—	—	—	PW_{11}	PW_{10}	PW_9	PW_8	PW HI
54290	NOISE				TEST	RING MOD	SYNC	GATE	CONTROL REG
54291	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK ⁄ DECAY
54292	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN ⁄ RELEASE
									FILTER
54293	—	—	—	—	—	FC_2	FC_1	FC_0	FC LO
54294	FC_{10}	FC_9	FC_8	FC_7	FC_6	FC_5	FC_4	FC_3	FC HI
54295	RES_3	RES_2	RES_1	RES_0	FILT EX	FILT 3	FILT 2	FILT 1	RES ⁄ FILT
54296	3 OFF	HP	BP	LP	VOL_3	VOL_2	VOL_1	VOL_0	MODE ⁄ VOL
									MISC
54297	PX_7	PX_6	PX_5	PX_4	PX_3	PX_2	PX_1	PX_0	POTX
54298	PY_7	PY_6	PY_5	PY_4	PY_3	PY_2	PY_1	PY_0	POTY
54299	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0	OSC3 ⁄ RANDOM
54300	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	ENV3

The noise waveform has no precise pitch; it's used for untuned sounds like percussion, wind, or jet engines. With this waveform, low notes come out as a deep rumble, and high notes sound like a snake's hiss.

The frequency of each tone generator is set by a 16-bit number that you POKE into two control registers (high and low bytes). The output frequency in Hertz is equal to the number in the registers multiplied by 0.0596. Table 1 gives the numbers that you use for the notes of the musical scale. SID has a range of eight octaves; is that enough for you? It ought to be; it's more than almost any conventional instrument can play.

With 16-bit frequency control, there are a lot of pitches in between the notes of the scale. You can generate glissando or portamento effects by rapidly incrementing or decrementing the number in these registers, so that the sound makes a smooth sweep from one pitch to another. You can also set two or three tone generators to be just a tiny bit out of tune with each other, which gives a rich, chorus-like quality to the sound.

Actually, SID's range is more than eight octaves: it can be tuned so low that you can't hear it, down to about 1 cycle every 16 seconds! We'll look at uses for the sub-audio range a little later.

Each voice has a control register that contains one bit for each of the four waveforms. If you turn on more than one of these bits, the resulting sound will be a logical ANDing of the selected waveforms. This could give you some interesting effects, but usually you will only use one waveform at a time. A word of warning: combining the noise waveform with any others may "lock it up", cancelling the noise output until you reset it with the Test bit or the chip Reset line.

When you select the pulse wave, the pulse width is set by a 12-bit number, which occupies two control registers. You can smoothly sweep the pulse width from one value to another, which gives a very nice soaring or "phase shifting" sound.

The control register contains bits for ring modulation and synchronization functions. These two effects are similar: they both take input from two tone generators, and produce an output that has some components of the inputs, plus some other frequencies that aren't present in either input. This can produce metallic sounds such as chimes and gongs. If you vary the frequency of one of the inputs while listening to the signal, you get a great science fiction-type sound in which you can hear some pitches rising, while others are falling at the same time. Note that the ring-mod function only affects the triangle wave output, but the synch function applies to all waveforms.

The difference between the two effects is something that I can't really describe in words, so I suggest you just try it. For some really wild sounds, you can use both effects at once. For example, you can set voice 2 to be in synch with voice 1, and set voice 3 to be ring-modulated by voice 2. I once did something like that with an ARP 2600 synthesizer, and got a really nice simulation of someone banging on a garbage can.

Each voice has a Test bit that, when set to 1, turns off all waveforms and resets the internal counters to zero. Commodore's spec sheets suggest that this feature may have some musical applications, but do not give any specific uses. I do have a couple of ideas, though: if you want to create a complex sound by combining two or three voices, you can use the Test bit to make sure that all the tone generators start their waveforms at the same moment; otherwise the slight delay might produce random variations in the sound. Also, the Test bit can be used to turn a voice on and off instantly, whereas using the envelope generator takes at least a few milliseconds.

Envelope Generators

This is another important synthesizer function. The term "envelope" refers to the way in which the volume changes during the playing of a note. Each note is divided into four phases called attack, decay, sustain, and release. In the attack phase, the volume rises from zero to a maximum or peak value. Then, during the decay, the volume falls off to some intermediate level. Next comes the sustain, in which the volume remains constant for as long as you want to hold the note. Finally, during the release, the volume falls back to zero.

The attack, decay, and release times, as well as the difference between the peak and sustain volume, are important factors in making one instrument sound different from another. For instance, as Figure 3 shows, a trumpet has a very short attack and decay time, giving a quick snap of loud sound at the beginning of each note. Then the volume remains constant as long as the musician keeps blowing, and when he or she stops, the note takes a tenth of a second or so to die out (release). Compare this to the violin envelope, which has a slow attack and no pronounced peak. The xylophone, by contrast, has a very fast attack, but no sustain at all; the note always dies away quickly.

Each voice in SID has its own controls for attack, decay, sustain and release. Each of these parameters is controlled by a four-bit number that can select one of sixteen possible values. The attack and decay are set by one control register, and the sustain and release by another. The attack times range from 2 msec. to 8 seconds. The decay and release times range from 6 msec. to 24 seconds. The sustain is not a time; it's a volume. If it is set to maximum (15), the volume will remain at the peak level, like the violin envelope in Figure 3.

The envelope generator is activated by a bit in the voice's control register called the Gate. (This is a synthesizer term, not really related to the logic gates we computer hackers are used to.) Setting the Gate to 1 starts a note; it causes the envelope generator to do its attack-decay-sustain phases. When the Gate is set to 0, the note begins its release phase. Note that the attack has a linear slope, but the decay and release have an exponential curve. This is a nice touch; it corresponds to the way that strings,

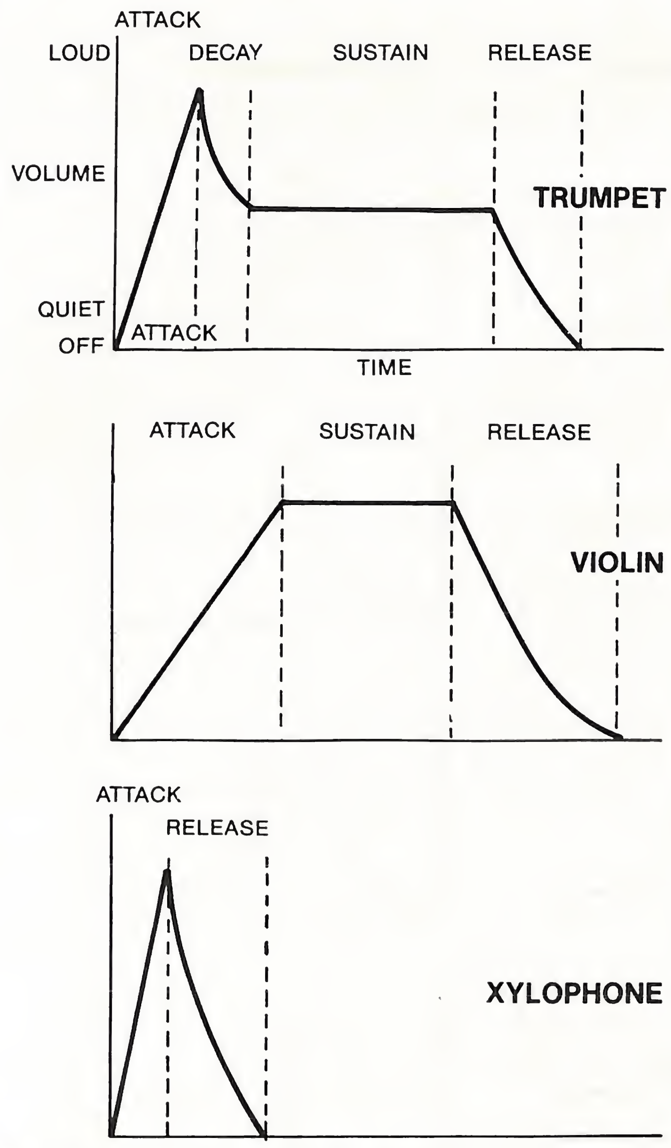
TABLE 1

Music Note Values

This contains a complete list of Note numbers, actual note, and the values to be POKEd into the HI FREQ and LOW FREQ registers of the sound chip to produce the indicated note.

Note	Note–Octave	Hi Freq	Low Freq
0	C–0	1	18
1	C#–0	1	35
2	D–0	1	52
3	D#–0	1	70
4	E–0	1	90
5	F–0	1	110
6	F#–0	1	132
7	G–0	1	155
8	G#–0	1	179
9	A–0	1	205
10	A#–0	1	233
11	B–0	2	6
12	C–1	2	37
13	C#–1	2	69
14	D–1	2	104
15	D#–1	2	140
16	E–1	2	179
17	F–1	2	220
18	F#–1	3	8
19	G–1	3	54
20	G#–1	3	103
21	A–1	3	155
22	A#–1	3	210
23	B–1	4	12
24	C–2	4	73
25	C#–2	4	139
26	D–2	4	208
27	D#–2	5	25
28	E–2	5	103
29	F–2	5	185
30	F#–2	6	16
31	G–2	6	108
32	G#–2	6	206
33	A–2	7	53
34	A#–2	7	163
35	B–2	8	23
36	C–3	8	147
37	C#–3	9	21
38	D–3	9	159
39	D#–3	10	60
40	E–3	10	205
41	F–3	11	114
42	F#–3	12	32
43	G–3	12	216
44	G#–3	13	156
45	A–3	14	107
46	A#–3	15	70
47	B–3	16	47
48	C–4	17	37
49	C#–4	18	42
50	D–4	19	63
51	D#–4	20	100
52	E–4	21	154
53	F–4	22	227
54	F#–4	24	63
55	G–4	25	177
56	G#–4	27	56
57	A–4	28	214
58	A#–4	30	141
59	B–4	32	94
60	C–5	34	75
61	C#–5	36	85
62	D–5	38	126
63	D#–5	40	200
64	E–5	43	52
65	F–5	45	198
66	F#–5	48	127
67	G–5	51	97
68	G#–5	54	111
69	A–5	57	172
70	A#–5	61	126
71	B–5	64	188
72	C–6	68	149
73	C#–6	72	169
74	D–6	76	252
75	D#–6	81	161
76	E–6	86	105
77	F6	91	140
78	F#–6	96	254
79	G–6	102	194
80	G#–6	108	223
81	A–6	115	88
82	A#–6	122	52
83	B–6	129	120
84	C–7	137	43
85	C#–7	145	83
86	D–7	153	247
87	D#–7	163	31
88	E–7	172	210
89	F–7	183	25
90	F#–7	193	252
91	G–7	205	133
92	G#–7	217	189
93	A–7	230	176
94	A#–7	244	103

Figure **3** *Sound envelopes of some typical instruments.*



horns, and other vibrating objects generally behave. Most synthesizers do either linear or exponential slopes, but not both.

Filter Section

The filter, in a sense, is the heart of a synthesizer. Granted, it doesn't actually produce sound, it just modifies what the tone generators produce. However, you will find that the filter has more control than anything else over what you hear. I'd rather have one waveform and a filter than a whole lot of waveforms and no filter.

The filter's function is similar to the tone controls on a hi-fi, in that it allows you to emphasize or remove certain parts of the audio spectrum. The range of possible effects is shown graphically in Figure 4.

SID's filter has three outputs. The low-pass output, as its name implies, will pass all signals below a certain frequency, called the cutoff frequency. Everything above the cutoff is "rolled off" (reduced in volume) at a rate of 12 dB per octave; the higher the frequency, the more it is reduced. Similarly, the high-pass output passes signals above the cutoff, and rolls off everything below it. The band-pass output rolls off frequencies above or below, and passes only those signals that are fairly close to the cutoff frequency.

Each of the filter's outputs is controlled by a single bit in one of SID's registers, so by setting several bits to 1, you can mix the outputs. Mixing the high-pass and low-pass outputs gives what is called a notch filter. This is the opposite of the band-pass: it rejects frequencies near the cutoff, and passes everything else.

The cutoff frequency is selected by an 11-bit number that you poke into two control registers. With the recommended 2200pf capacitors, the cutoff frequency can range from 30Hz to 10kHz.

The filter also has a resonance or "Q" control. This determines how strong the effect is. With low resonance, the sound is not too different from your hi-fi's tone controls. With high resonance, you get a very intense effect, like different vowel sounds of the human voice. A rock guitarist's wah-wah pedal is just a band-pass filter with a very high resonance. It goes "wah" when the musician raises the cutoff frequency by pushing down on the pedal, and it goes "yow" when he or she lowers the cutoff by moving the pedal the other way. You can produce the same effect with SID by selecting a high resonance, and varying the cutoff while a note is playing.

Most synthesizers provide an envelope generator that controls the filter cutoff, allowing it to automatically rise and fall every time a note is played. SID does not provide this feature, but you can write a program to do it. Some of the more expensive synthesizers can also change the resonance in the same manner. The effect is more subtle, but useful to the advanced synthesist. With SID, once again, you can do this under program control; isn't software wonderful?

Each of the three voices can be routed through the filter, or it can be sent directly to the main output. This feature helps make up for the fact that there are three voices but only one filter. You can filter one voice, and use pulse width changes to produce filter-type effects on the other two.

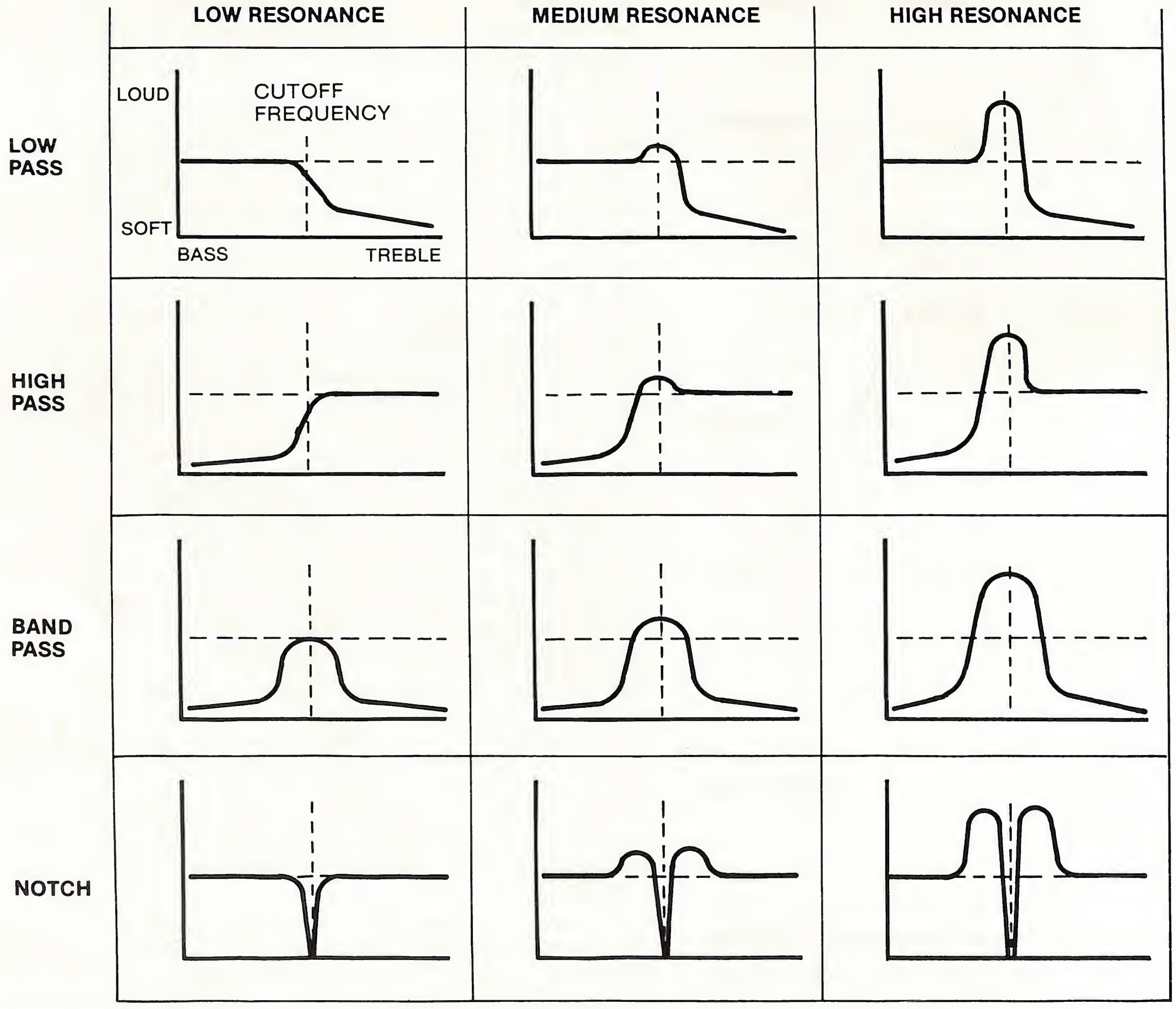
Other Features

As I mentioned earlier, SID has an external input that can be used with any instrument, recording, or even a microphone. This input can be sent through the filter, or it can go directly to the output. With a microphone or instrument, you may need to amplify the signal before running it into SID. The maximum input is 3 volts peak-to-peak.

SID has a master volume control that ranges from zero to 15. This controls all the voices, the filter, and the external input. You can produce tremolo effects by rapidly raising and lowering the volume.

Voice 3 has a couple of special features that are quite handy. There is a register that you can read to get the

Figure **4** *Some effects that SID's filter can produce.*



instantaneous value of the tone generator's output, and another that lets you read the envelope generator's output. What are these good for? Well, earlier I mentioned that it would be nice to have another envelope generator to control the filter. You can use voice 3's envelope generator to do this, by putting some statements in your program to read the envelope value, and POKE it into the filter cutoff.

You can use the waveform value register in a similar manner. For instance, set up voice 3 to produce a triangle wave at a very low frequency, say 3 or 4 Hz. Then write a program that continuously reads the waveform value, and adds it to the frequency numbers for voice 1 and/or 2. The result is vibrato: continuous up-down variation of the pitch.

Of course, when you're using voice 3 in this manner, you probably won't want to listen to it at the same time. Fortunately, there is a control bit that disables voice 3's output, so that it won't produce unwanted sounds.

One other use for voice 3: if you set it to the noise waveform at some very high frequency, the waveform value at any instant is essentially random. Thus you have an instant random number generator.

SID contains two A/D converters that are intended for connection to pots, for use as game paddles or other control functions. You might want to go to a music store, and buy an old used wah-wah pedal. Tear out all the electronics, put in a 470K pot, and run it to one of the A/D inputs. Presto, you've got a wah-wah pedal for your computer . . . or a volume pedal, or a tremolo pedal, etc. Like I said, isn't software wonderful?

The A/D inputs are scanned about once every half a millisecond, so it's possible that they could be used to listen to an external sound and compute its pitch. Then SID could play or sing along with you.

Technical Details
Originally I was going to title this section "Bugs" or "Criticisms", but really, this chip is so wonderful that I couldn't bring myself to use such derogatory language. However, there are one or two things that I just couldn't resist pointing out.

Most of the control registers are "write-only," meaning that you can POKE things into them, but you can't PEEK at them to find out what their current settings are. So you may need to store their values in a table, and have your program update the table whenever it changes one of the registers.

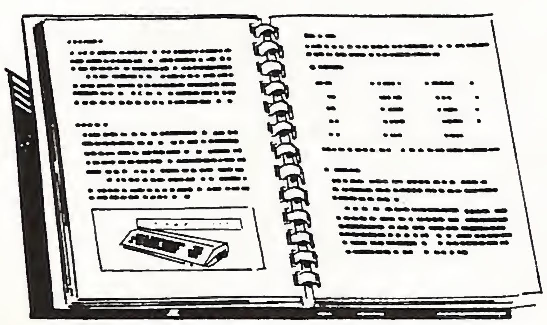
If you've used some other synthesizers, you may notice that SID's filter effects are not quite as intense as you expected. This is because SID's filter has a 12 dB per octave roll-off, whereas most synthesizers have 24 dB per octave. However, you can always set two SIDs, and run one into the other through the external input. It'll still cost a heck of a lot less than a new Korg or MiniMoog.

Although SID has a master volume control, it does not have controls that let you set one voice louder or quieter than the others. This could be a problem if you're trying to play a fairly complex piece of music with a lot of variety between the voices. In a pinch, you could write a program to rapidly switch a voice's Gate signal on and off, so that the volume hovered around some intermediate level.

In summary, my hat is off to Commodore for producing a chip that turns a home computer into a real synthesizer. The musical world is bound to be rocked by this exciting new development. Does anyone out there want to buy a used Korg?

Kent Multer *(P. O. Box 732, W. Acton, MA 01720) wrote his first computer game in 1970, at the age of 14. He later attended Worcester (Mass.) Polytechnic Institute, where he did his first experiments with computer music. He is now a free-lance author and programmer.*

Extensive Software for the Commodore 64



Programmers Reference Guide for Commodore 64

A 486 page PROGRAMMERS REFERENCE GUIDE has been released for the COMMODORE 64. The manual retails at $19.95 in the U.S.A. and provides the user with what could well be the most comprehensive reference source available on any personal computer today. It includes everything from MOS Technology chip specifications to memory maps and is an indication of the increased importance Commodore has been placing on user documentation.

Each section of the book was written, before editing, by the Commodore expert who worked most extensively in that area. It was then tested at nearly 20 different Beta sites for accuracy and understandability. The informaton provided is repeated on two different levels, one for the novice and one for the expert, with all sections color highlighted for easy reading.

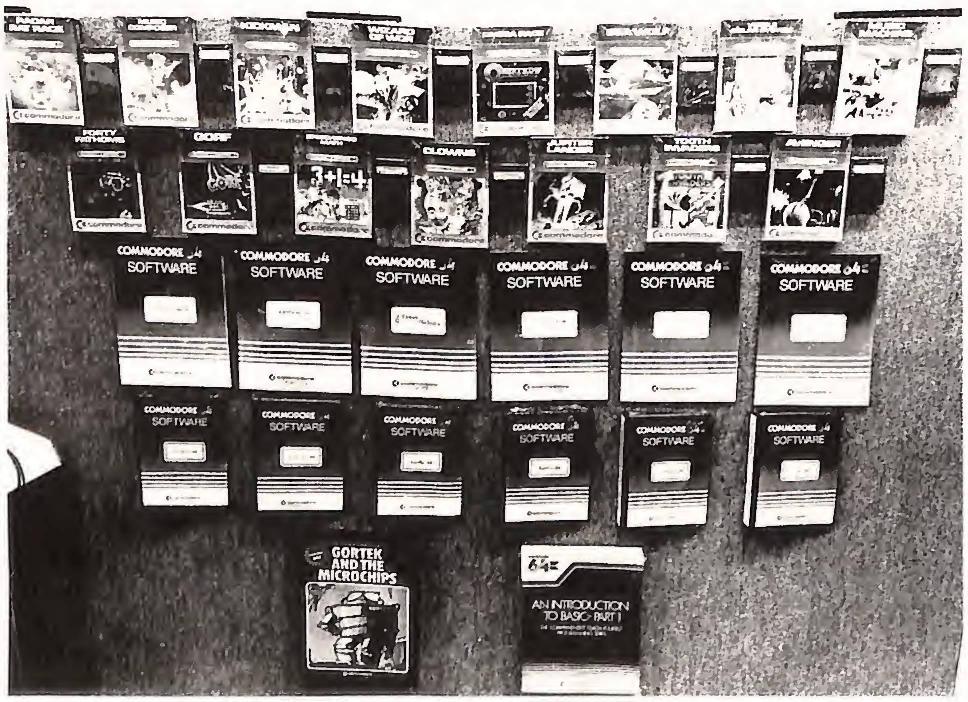
This Reference Guide is the first in a series of manuals being prepared for users of the popular COMMODORE 64 computer. Other titles planned for release in early 1983 are: MAKING MUSIC ON THE COMMODORE 64, CP/M FOR THE COMMODORE 64, and INTRODUCTION TO BASIC PROGRAMMING.

In addition to distribution through Commodore dealers and retailers the Reference Guide will be distributed through bookstores via an agreement with the Howard Sam's book publishing company.

Released at the C.E.S. show was a 12 page catalog of software offerings for the Commodore 64. This catalog represents a comprehensive range of programs planned at the time of the development of the 64 and coordinated via Commodore companies with software houses worldwide. This swift coordination on an international basis could only be done by a company like Commodore International.

In addition to products developed in the U.S.A. major contributions came from the U.K., particularly in the business and programming aid areas, and from Japan in the games field. A large number of "public domain" educational programs were put together in Canada. Where appropriate, these programs will be offered for sale on a worldwide basis through Commodore companies and distributors who will make any necessary translations required by local market conditions.

The range of software has been classified into seven major sectors of interest. A few of the programs were put on sale in the first quarter of 1983 while most of the balance of more complex programs have been undergoing final testing for release during the second quarter of 1983.



Business Aids

EasyCalc

- One of the largest electronic worksheets available.
- Selective row reporting and printing.
- Instant "what if" type calculations.
- Integration of spreadsheets, from disk to memory, allows summation and consolidation of different size and type matrices.

EasyFile

EASYFILE is a database product that allows the user to define the data he wishes to keep track of, how it looks on the screen, and the ability to enter and retrieve that data with editing and selective criteria.

- Simple instructions and "on line" help functions.
- Individual field editing on entry to prevent bad data from being entered into the system.
- Ability to add, change, delete, report on and search for the data in any manner required.
- The program is memory resident, so the entire disk is free for data.
- Sorting of the data may be accomplished with up to eight levels of dependence.

EasyMail

A fully featured name and address program for the serious user that requires a complete name address system for a small business, club or organisation.

- Allowance of entry/change/deletion of a name and address by number or name.
- One or two abreast address labels.
- A complete printout of the system is possible showing all information.
- The capability to search and sort any field for creating a special list or mailings.
- A "help" function that is available at all times.

EasyPlot

- Full page printing of charts and graphs.
- Integration up to eighty data points directly.
- Will plot up to four data series at a time, depending on the type of plot required.

EasySchedule

- The ability to tailor certain areas of the program to your specific scheduling needs.
- Input of data in easy logical steps.
- Day, Week, Month, and Year at a glance lets you see the scheduled items by time slots.
- The ZOOM option lets you zero in on smaller and smaller time slots.
- Organization of data by priorities.

EasyScript

In concept Easyscript is very similar to Zardax, one of the Apple's more popular wordprocessors.

- Screen text width from 40 to 240 columns.
- Supports non–Commodore printers.
- Total and complete cursor movement.
- Function keys for frequently used options.
- Vertical and horizontal tabs.
- Range selection for text movement options.
- Search and replace.
- Output to printer, disk, "fill" files as well as video for checking formatting.
- Scroll left, right, up and down.
- Supports user defined characters.
- Underline, bold, shadow, super/subscript, on supported printers.
- Right margin justification or alignment.

EasySpell

Do you ever misspell words? Wouldn't it be nice if you had a computerised spelling checker that automatically corrected your errors, counted the words in a manuscript or contract? Wouldn't it be nice if the spelling checker had a 20,000 word spelling dictionary built in, and was totally compatible with your wordprocessor? What if you could add your own "frequently misspelled words" to the dictionary? That's EASYSPELL 64! This ingenious disk–based program is designed to work with your EASYSCRIPT 64 wordprocessor. As soon as you use it, you'll say, "What a terrific idea!"

EasyStock

EASYSTOCK is a comprehensive stock control system created specifically for retailers and smaller businesses. The reason it's "EASY" is because you don't have to be a computer expert to set up and operate the system. Simply enter your stock records and store the information on convenient floppy diskettes. Items include stock description, location, reorder level, tax rate, profit margin and cumulative/purchase totals, to name a few. Also generates printed reports on stock levels, stock movement, sales returns (weekly or annual basis), stock–taking lists, valuation reports and more.

Word/Name Machine

This product is our most easy–to–understand word processing product, and is designed as an entry level product for the home. It is perfect to jot down a note to the kids, letters to friends and so on.

- Simple, easy to understand menu operations.
- Allows overtype, insert and delete of text in an easy to understand "building block" approach.
- Three print formats, draft, informal and formal.
- Will generate personalised form letters via a link to Commodore Name Machine, its companion program.
- Simple creation of a name and address list.
- Prompts are given for Last, First, Title, Organization, Address, State and Zip, Telephone and Category.
- Can generate a Name and Address file to be used in form letter processing in the Word Machine.
- Can select names and addresses by category.

Learning Aids

Introduction To Basic— Parts I And II

By the time you're done with this two–part self–teaching course, you'll be writing advanced programs like an expert! Just LOAD the cassette tapes and follow the word books. The course is divided into individual units each of which covers one aspect of BASIC programming. Two cassette tapes in each Part provide sample programs, and questionnaires are also included. This time–tested programming course has become a "classic" in the United Kingdom where it was developed, and the VIC–20 version is a best–seller in Australia and the United States.

Gortek and the Microchips

Can you help GORTEK stop the Zitrons! This delightful and unique concept in teaching young people to program combines a space adventure story with lessons in BASIC programming. The first lesson consists of 2 tape cassettes and a colorful glossy instruction book which reads like a comic book while teaching the fundamentals of BASIC programming. The book includes full color instruction, large easy–to–read type and is written so it may be read by older children... or used by younger children with parental assistance. The combination of computer lessons and storybook format makes GORTEK a fun experience for adults as well as children.

Programming Aids

Assembler 64

- Allows the experienced user to write software in assembly language.
- Allows the total extraction of the machine power and function while developing applications that run at maximum speed.
- The package contains the assembler itself, two machine code monitors, editor, DOS wedge, and two loaders.

Logo

Commodore has contracted with the original authors of this favorite graphics language to put it in on the Commodore 64. The adaption will be complete and similar to the Apple LOGO by Terrapin, plus extensions that really show off the power of the sprite graphics on our machine.

- Graphics definition and movement.
- Text display.
- Multi–level graphic character display.
- Easy–to–use instructions. Uses the Commodore defined keyboard for screen editing.

Pilot

- All the features defined as "Common Pilot".
- Interfaces for Video disk/tape players to help get into the industrial education market.
- Because the adaption is based around Common Pilot, certain lessons that are already in the field may be modified to run under our PILOT with little change.

Pet Emulator

An emulator that will allow a high level of existing BASIC PET (2001) software to be executed on the Commodore 64.

- The ability to recapture the tremendous amount of software that has been generated on the PET product, especially in the educational area.
- Effective translation of an existing program commands to run on a Commodore 64 as if it were written for it.

The effort to create this emulator displays our intention to have everyone benefit from existing software. The Commodore 64 is a quantum leap ahead of the PET in design and function, but we will not ignore the existing software base.

Simon's Basic

It's called "SIMON'S BASIC" because it was developed by a programmer named SIMON who wanted to add 114 extra commands to BASIC... so he did. Commodore put it on a cartridge and now you can take advantage of four complete GROUPS of commands: HIGH RESOLUTION (to draw shapes on the screen), STRUCTURED PROGRAMMING (to write more legible BASIC code), MUSIC COMMANDS (for musical composition), and PROGRAMMING AIDS for easy program writing and editing. Examples include IF... THEN... ELSE, REPEAT... UNTIL, LOOP... EXIT... to name a few. AUTO gives you automatic line numbering. KEY makes it easy to define function keys. If you're a programmer, you're going to love SIMON'S BASIC!

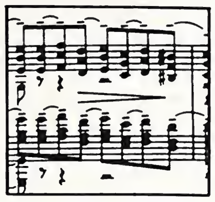
Art and Music Series

Music Machine

Use the sound capabilities of the Commodore 64 and turn your computer keyboard into a musical keyboard. Create special effects, waveforms, percussion, octaves and pitches.

Music Composer

Compose a song, play a song, load the song and save the song. Turn your Commodore 64 into a piano keyboard, choose your instrument and play your song.



Meta Music

A music tutorial that allows the novice, as well as the serious composer, to create music on the Commodore 64.

- Displays all phases of the computer's sound capabilities in detail.
- Demonstrates examples of tempo, harmony, and composition and examines them from the programmer's point of view, as well as practical useage.
- Is disk and RS 232 printer compatible.
- Has two modes of operation: SCORE and PLAY.
- Will implement various music education processes (e.g. note and interval identification, sight singing, melody recognition, rhythm training, and elementary timbre identification.

RENAISSANCE
GAME CARTRIDGE FOR THE COMMODORE VIC 20
SPIDERS OF MARS
GAME CARTRIDGE FOR THE COMMODORE VIC 20
ALIEN BL
AMOK
GAME CARTRIDGE FOR THE COMMODORE VIC 20
METEOR RUN
GAME CARTRIDGE FOR THE COMMODORE VIC 20
SATELLITES & METEORITES
GAME CARTRIDGE FOR THE COMMODORE VIC 20
umi

UMI software...a world of choices

A World of Fun! They're hot! They're new! The exceptional graphics and challenging play of UMI's games have made United Microware the leader in arcade-quality recreational software.
A World of Help! UMI has created programs to help professionals and homeowners "take care of business." UMI can make your life a little easier with word processing, information storage, financial management, hobbyist programs, utilities and communication programs — all with easy-to-understand instructions.
A World of Choices! All programs come on cassettes or UMI's own durable cartridges, depending on your selection. If you're looking for fun, or for an easier way to manage your personal business, look to UMI . . . the leader you can trust. UMI products are available at your favorite computer products store.

umi

Available from all fine microcomputer stores and at Dick Smith Stores.

APPROVED SOFTWARE
IMAGINEERING
3/579 Harris Street, Ultimo, NSW 2007
Telephone 212 1411

Contact Imagineering for your nearest microcomputer dealer
NAME ....
ADDRESS ....
PROGRAMME NAME ....

PPCB1488APCE

Software Reviews
by Dirk Williams

VIC TEXT REVIEW

VIC Text is a new product produced by The Microcomputer House for the VIC 20. The VIC Text cartridge also contains an 8K of extra RAM, meaning that the editor can still be used effectively on an unexpanded VIC. The general impressions that VIC text gave were very good.

VIC Text is a wordprocessor like editor that can be used for editing text files, programs, etc. It contains a full complement of functions including global search and replace, block move, delete, making it a very powerful package. The editor is full screen editor from which any part of the text may be viewed – like a window. Text may be stored on tape or disk, printed with left hand justification, if required.

Operation is fairly simple once one gets used to the many control codes (accessed by pressing the STOP key and then the key of the function you want) and "ESC" functions. There is a comprehensive error reporting system which eases operator mistakes. Unfortunately the package lacks user friendliness –however, considering the number of features they have put in a small package, who's complaining?

The system is provided with a comprehensive manual. I feel that the manual would be vastly improved with the addition of a few examples.

Overall an excellent package, well worth considering. Any further enquiries should be directed to:

The Microcomputer House,
Telephone: (02) 698 7076.

ACME SOFTWARE

Mr Joel Gotlib of Edible Electronics in Melbourne has just started a software company aimed at the VIC 20. It is believed that he has a number of creative programmers. With a company name like ACME how can it fail.

We were lucky enough to get some of the first copies of the present range. The three packages we were given to look at were VIC DERBY, VIC VOICE, VIC SENTINEL and LOCOMOTION. Both of these packages require either an 8K or 16K memory expansion cartridge. The cassettes are produced in similar packaging to the Commodore games cartridges. The whole package looks neat and tidy. Joel seems to be very keen to promote the fact that all the software is produced in Australia, because the message "AN AUSTRALIAN PRODUCT" is plastered all over the packaging.

Something that I thought was very interesting was the wording of warranty. It states very clearly that the product is warranted against malfunction due to faulty materials or workmanship for 90 days from date of purchase. This should avoid the situation of buying the software and finding that it does not work once it is at home.

VIC DERBY

I do not know if you have ever seen the Ye Old game of Horse Races on the CBM, where little figures run across the screen that represent horses. Anyway VIC DERBY is very similar. What the program does is to allow one to place bets on the horses before they run. The horses have allocated to them a probability figure which corresponds to the chance of that horse winning.

The game will handle up to 8 players, but does not have the facility to have 0 players so that it plays by itself, drat. Despite this fact the game is very entertaining. Betting is quite straight forward, you have the option of betting "each way" or "to win" or for "a place". Money will then be paid out on that basis.

LOCOMOTION

This is one of the most interesting VIC games I have seen for some time. The concept of the game is really what makes the game so good.

The object of the game is to keep an ever increasing number of trains in play. A railway network is displayed on the screen, the interweaved lines are connected by points. By pressing a number – which corresponds to a particular point – it is possible to change the line to which the point is connected.

By arranging all the points one can construct a loop for the trains to flow along. This is fine until a train going in the opposite direction to your loop is introduced. This then forces the system to be reorganised. Once a full complement of trains are going it becomes quite a feat to contain all the trains.

P.S. The sound effects on this game are very amusing.

VIC VOICE

An excellent package for the experimenter and programmer. It allows the VIC to "play" music and voice through its standard sound channels. Sound can be stored in the memory and inputted via the Commodore Datasette in record mode – this digitizes the sound input and allows playback at a later date from the memory. In cue mode, the VIC interprets and plays sound from the datasette – through its sound channels without storing in the memory. The code is also designed so that programmers can utilize its playback in their own programs –to add a voice perhaps. All over, a good program. Although the sound produced was generally a bit poor but easily understandable.

Well done Acme.

SENTINELS

This program has a good presentation and screen effects, however tends to become a bit tedious after a while. While it is an ideal game for youngsters – it would soon bore adults. This program would be vastly improved if the "path" upon which you must travel, avoiding sentinels, were wider and the sentinels followed you in higher levels of play.

Record Second Quarter Results

Mr. Irving Gould, Chairman of the Board of Commodore International Limited announced on January 25 that Commodore had achieved record sales, net income and earnings per share for the second quarter and six month period ended December 31, 1982

In commenting upon these results, Mr. Gould stated that "Commodore's record results in the second quarter and six month period resulted from continuing very strong sales of Commodore's microcomputer systems to the business, educational, personal, and home markets."

Mr. Gould went on to say that "Commodore expects to start shipping its new $795 COMMODORE 128 (P500 series) personal computer to its dealer network this quarter, while its new B700 and BX700 small business microcomputers would commence shipment during the final quarter of fiscal 1983 which ends June 30th. Both the B700 and BX700 microcomputers will be priced substantially below any other small business microcomputers with comparable features now on the market and will be sold exclusively through Commodore dealers in the United States and abroad."

"In addition," Mr. Gould noted, "Commodore will also begin shipping its brand new 64K portable color computer in approximately 120 days, this being a brand new market opportunity for Commodore."

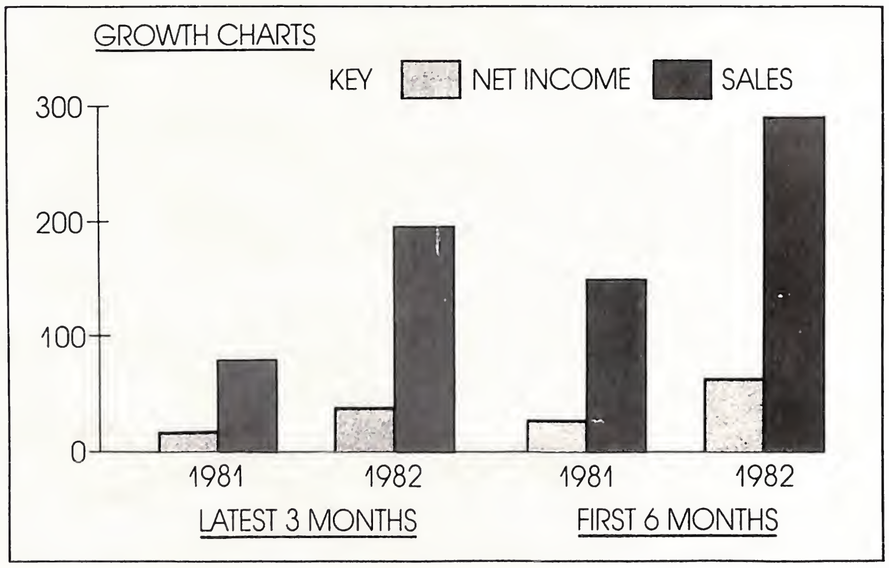
Mr. Gould concluded by stating that "with the current strong demand for our small business, educational, personal, and home computers, fiscal 1983, which ends in five months, should result in our sales, net income and earnings per share being significantly ahead of the record results we reported in fiscal 1982."

**COMMODORE INTERNATIONAL LIMITED
AND SUBSIDIARIES
CONDENSED STATEMENT OF OPERATIONS
$(000)**
(unaudited)

	Three Months Ended December 31,		Six Months Ended December 31,	
	1982	1981	1982	1981
Net Sales	$176,241	$70,056	$279,571	$124,206
Income from Operations	29,314	11,065	46,144	20,675
Provision for Income Taxes	6,345	2,300	9,825	4,090
Income before Extraordinary Item	22,969	9,305	36,319	16,585
Extraordinary Item (A)	2,080	—	3,680	300
Net Income	$ 25,049	$ 9,305	$ 39,999	$ 16,885
Earnings Per Share (B)				
Before Extraordinary Item	$ 1.49	$ .61	$ 2.36	$ 1.08
Extraordinary Item (A)	.14	—	.24	.02
Net Income Per Share	$ 1.63	$ .61	2.60	$ 1.10
Average common shares and common share equivalents	15,408,000	15,352,000	15,384,000	15,402,000

(A) Represents tax loss carryforward benefit.
(B) 1982 per share amounts have been restated to reflect a 3-for-2 stock split in 1982.

Commodore International Reports Record Second Quarter Financial Results



JOIN THE CLUB

AND GET MORE OUT OF THE VIC CENTRE CATALOGUE

★ Discounts *
★ Exchange Commodore Cartridge Service
★ Membership Specials

GET YOURS NOW!

MORE THAN 30 PAGES PACKED FULL OF GOODIES

1983 AUTUMN VIC CENTRE CAT.

YOU DON'T HAVE TO BE A CLUB MEMBER TO GET THE MOST EXPANSIVE CATALOGUE OF COMMODORE VIC & C64 SOFTWARE & HARDWARE AVAILABLE IN AUSTRALIA *BUT IT PAYS.*

INCLUDES COMMODORE'S OWN AND LATEST VIC 20 & C64 RETAIL PRICE LISTS

WRITE FOR ONE TODAY

★ **CLUB MEMBER BONUS;**
MEMBERSHIP DISCOUNT: For every one dollar spent with the VIC Centre you will receive a credit certificate for 12 cents against future purchases e.g. Commodore 64 $699 gives Credit Certificate of $83.88, etc. This does not apply to orders below $50, Club Specials, items purchased using Credit Certificates, or on trade-ins.

THE CLUB

The VIC CENTRE COMPUTER CLUB APPLICATION
CW Electronics 416 Logan Road, Stones Corner, Q'ld.
P.O. Box 274 Sunnybank Q'ld 4109

NAME: ....

ADDRESS:.......................................

....

SUBURB: .... P/CODE:............

PHONE: ....

Membership Fee $55.00/$30.00

☐ Pay by Bankcard/
Bankcard No: ____________
Expiry Date: ______

☐ Pay by Cheque/Money Order

Cartridge Title required: ____________
(VIC-20 only at this stage)

Alternative ____________

Signature:(Member and Card Holder) ....

RECEIPT & RULES

CONDITIONS OF MEMBERSHIP:

(1) Membership fee of $55.00 (incl. first Commodore cartridge) Non-cartridge member $30.00, both renewable annually. $10.
(2) Reduced membership rates are available with additional cartridges purchased at the time of making membership application.
(3) Exchange of Cartridges allowed at the rate of $7.00 at time of exchange. (+ $1 certified postage if applicable).
(4) The variety and number of cartridges available for exchange will be controlled by The VIC Centre.
(5) All Cartridges returned for exchange must be working and undamaged.
(5A) A service charge will be incurred on all repairable cartridges.
(6) Packaging and instructional material must be returned with the cartridge and in good condition.
(7) The cartridge returned must be the same cartridge borrowed.
(8) MAXIMUM time exchange is 1 calendar month, after this time the cartridge is unreturnable and deemed the property of the member.

Date: / /1983.

SIGNATURES:(Client) ....

(The VIC Centre) ....

NOTE: The signing of this form indicates agreement to the rules of membership.

AARDVARK (USA)

Adventure games for the VIC-20

16K expansion required

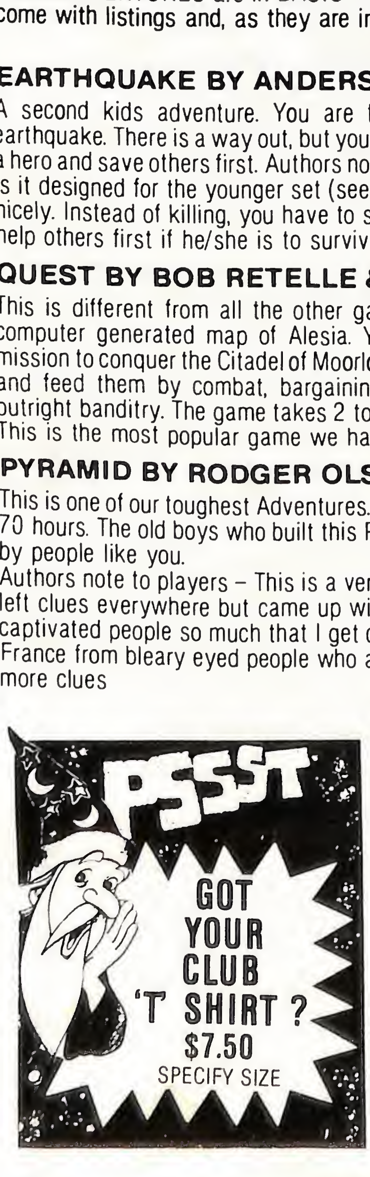
Available soon for the Commodore-64

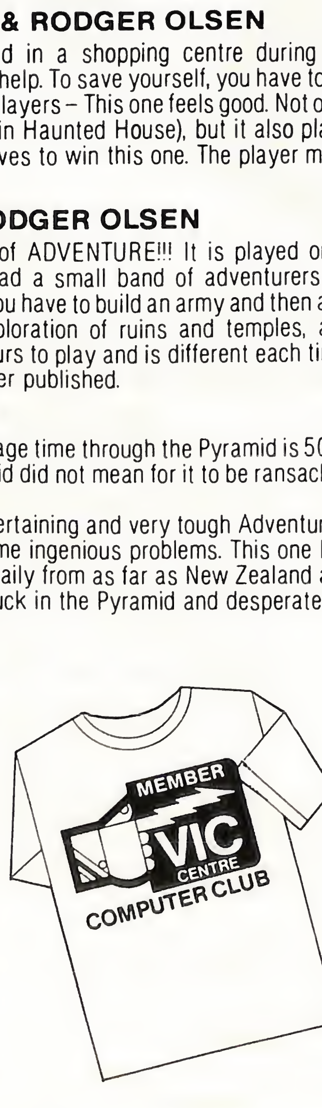
ADVENTURES are interactive fantasies. It's like reading an exciting book, except that you're one of the characters. You explore a new world as you try to think or fight your way out of a jam. You give the computer plain English commands such as "look in coffin" and "light the torch" and it carries out your bidding. Each ADVENTURE normally takes from 15 to 30 hours to play, spread out over several days. If the FDA ever catches us, we are going to have to add a warning label. These are definitely addictive!!!
These ADVENTURES are in BASIC – but they are full featured, full plotted, fast action adventures. They come with listings and, as they are in BASIC, you can modify them yourself.

EARTHQUAKE BY ANDERSON & RODGER OLSEN **$ 20.00**
A second kids adventure. You are trapped in a shopping centre during an earthquake. There is a way out, but you need help. To save yourself, you have to be a hero and save others first. Authors note to players – This one feels good. Not only is it designed for the younger set (see note in Haunted House), but it also plays nicely. Instead of killing, you have to save lives to win this one. The player must help others first if he/she is to survive.

QUEST BY BOB RETELLE & RODGER OLSEN **$ 20.00**
This is different from all the other games of ADVENTURE!!! It is played on a computer generated map of Alesia. You lead a small band of adventurers on mission to conquer the Citadel of Moorlock. You have to build an army and then arm and feed them by combat, bargaining, exploration of ruins and temples, and outright banditry. The game takes 2 to 5 hours to play and is different each time. This is the most popular game we have ever published.

PYRAMID BY RODGER OLSEN **$ 20.00**
This is one of our toughest Adventures. Average time through the Pyramid is 50 to 70 hours. The old boys who built this Pyramid did not mean for it to be ransacked by people like you.
Authors note to players – This is a very entertaining and very tough Adventure. I left clues everywhere but came up with some ingenious problems. This one has captivated people so much that I get calls daily from as far as New Zealand and France from bleary eyed people who are stuck in the Pyramid and desperate for more clues





MORE ADVENTURES!!

MARS BY RODGER OLSEN
Your ship crashed on the Red planet and you have to get home. You will have to explore a Martain City, repair your ship, and deal with possibly hostile aliens to get home again. Authors note to players – This is highly recommended as a first adventure. It is in no way simple playing time normally runs from 30 to 50 hours – but it is constructed in a more "open" manner to let you try out adventuring and get used to the game before you hit the really tough problems. **$ 20.00**

CIRCLE WORLD BY BOB ANDERSON
The alien culture has built a huge world in the shape of a ring circling their sun. They left behind some strange creatures and a lot of advanced technology. Unfortunately, the world is headed for destruction and it is your job to save it before it plunges into the sun!! Editors note to players – In keeping with the large scale of Circle world, the author wrote a very large adventure. It has a lot of rooms and a lot of objects in them. It is a very convoluted, very complex adventure. One of our largest. **$ 20.00**

DERELICT BY R. OLSEN & B. ANDERSON
For Wealth and Glory you have to ransack a thousand year old space ship. You'll have to learn to speak their language and operate the machinery they left behind. The hardest problem of all is to live through it. Authors note to players – This adventure is the new winner in the "toughest Adventure at Aardvark Sweepstakes". Our most difficult problem in writing this adventure was to keep it logical and realistic. There are no irrational traps and sudden senseless deaths in Derelict. This ship was designed to be perfectly safe for it's builders. It just happens to be deadly to alien invaders like you. **$ 20.00**

VAMPIRE CASTLE BY MIKE BASSMAN
This is a contest between you and OLD DRAC – and it's getting a little dark outside!!! **$ 20.00**

HAUNTED HOUSE BY BOB ANDERSON
This one is for the kids. The house has ghosts, goblins, vampires and treasures – and problems designed for the 8 to 13 years old. This is a real adventure and does require some thinking and problem solving – but only for kids. Authors note to players – This one was fun to write. The vocabulary and characters were designed for younger players and lots of things happen when they give the computer commands. This one teaches logical thought, mapping skills, and creativity while keeping their interest. **$ 20.00**

INTERCEPTOR MICRO'S (UK)

FROG
An amazing version of Frogger in the unexpanded VIC-20. With driving turtles and plenty of vehicles to run you down. Fast action and high resolution graphics. **$ 18.00**

PUCKMAN – MACHINE CODE
The old favourite back again. Joystick or keyboard control. Fast action. High resolution colour graphics on the unexpanded VIC 20 **$ 18.00**

GALAXZIONS – MACHINE CODE
This is the most amazing game ever seen on the VIC-20. Galaxzions swarming in attack formation to destroy your planet. The nearest program to the real arcade game for the unexpanded VIC-20 **$ 18.00**

WORDHANGER
A highly educational hangman game with vocabulary and 2 player or play against the computer option **$ 18.00**

VIC BOMBER
An extremely fast action Bomber game for the VIC-20. With high resolution colour graphics. Flatten the enemy city before it's too late. **$ 18.00**

STACK (UK)

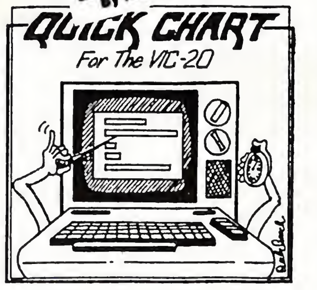
LIGHT PEN
plus 1 game additional games available **$ 70.00**
ANALOGUE JOYSTICK **$ 40.00**

THE VIC CENTRE
In conjunction with CW Electronics
416 Logan Road.
Stones Corner Brisbane
4120 P.O. Box 274 Sunnybank
Q'ld 4109. Australia
Tele: (07) 397 0808 397 0888
Telex: AA40811

ABACUS SOFTWARE (USA)

ALL ABACUS software is available on DISK or CASSETTE and most also include demonstration programmes.

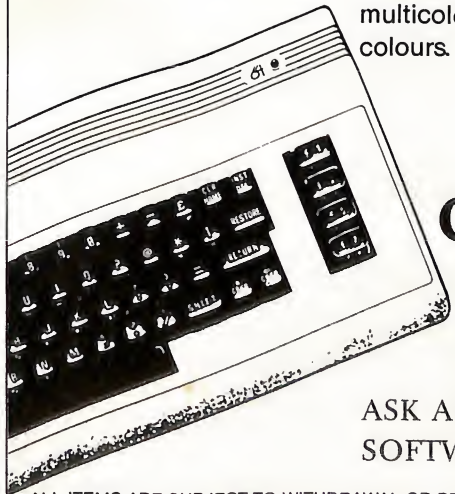
SCREEN-GRAPHICS-64 – Super graphics for the COMMODORE-64 Here's the finest graphics software for your COMMODORE-64. SCREEN-GRAPHICS-64 gives you High Resolution, Multicolour and Sprite graphics all in one package! You get 320 x 200 points in High Resolution mode, 160 x 200 in Multicolour and Sprite Graphics in either HiRes or Multicolour modes. This is the most powerful package for the COMMODORE-64 yet developed. SCREEN-GRAPHICS-64 gives you two screens – one for normal text and the other for graphics displays. You can switch back and forth between the two screens in either command mode or from another program control mode. And you can mix both HiRes and Multicolour and Sprites on the same screen.
$28 cass
$31 disk



QUICK CHART for the VIC-20 and Commodore-64 – QUICK-CHART is a tool for creating bar charts on either your VIC-20 or COMMODORE-64 (*VIC-20 requires 8K expansion*). It is designed to give you presentation quality charts from the data that you enter at the keyboard. QUICK-CHART is menu-driven which makes it very easy to use. QUICK-CHART prompts you for data and then edits the data you enter against your minimum and maximum values. It does automatic scaling and if you wish to create monthly bar charts it does automatic prompting. Whats more is that you can save up to 10 charts onto either cassette or disk (3 charts with the VIC-20) so that you can later recall them for redisplay.
$ 24.00 cass
$ 27.00 disk

PIPER – The Music Machine for your VIC-20 or PET/CBM micro PIPER is an exciting way to turn your VIC or PET/CBM micro into a music machine. PIPER lets you compose, conduct and play your own musical scores with ease. And you'll be amazed at the speed with which you can enter your music using PIPER. PIPER works on the VIC-20 and PET/CBM micros, including the 8032.
$ 24.00 cass
$ 27.00 disk

HIRES/MULTICOLOUR GRAPHICS for the VIC-20 – The VIC has two additional graphic display modes available for anyone to use. Here's a package that gives you immediate access to these impressive VIC features. The HIRES/MULTICOLOUR GRAPHICS utilities make it simple and inexpensive to add these graphic display capabilities to your VIC. And best of all they require no additional memory. Without any additional hardware the high resolution utility provides a display area of 104 x 152 plot positions. You can draw and or erase points, lines, rectangles or write text onto the HiRes screen. The multicolour utility offers 52 x 76 plot positions in multiple colours.
$ 24.00 cass
$ 27.00 disk



WE HAVE NEW C64 PROGRAMMES COMING IN ALL THE TIME

ASK ABOUT ANY SPECIFIC PET/CBM SOFTWARE OR HARDWARE

ALL ITEMS ARE SUBJECT TO WITHDRAWAL OR PRICE AND SPECIFICATION CHANGES WITHOUT NOTICE

SPRITE-AID for the COMMODORE-64
HiRes or Multicolour graphic were never this easy!
$ 24.00 cass
$ 27.00 disk

SYNTHY-64
Music and Sound Synthesizer for COMMODORE-64.
$ 33.00 cass
36.00 disk

SKIER-64 for the Commodore 64
Test your skill on the slopes with SKIER-64, a fast moving game for your CBM-64. This arcade-quality game requires good hand-eye coordination for you to score high. And you can play without even getting cold in the snow.
$ 24.00 cass
$ 27.00 disk

BUDGETEER for the VIC-20, COMMODORE-64 or PET/CBM
Now you can use your VIC-20, CBM-64 or PET/CBM to plan and track your personal budget for the year with our visual planner BUDGETEER. Its features are surprisingly wide including bar chart and printout options
BUDGETEER requires a VIC with a *3K, 8K or 16K memory expander, a* PET/CBM *with at least 8K of memory or a* COMMODORE-64.
$ 24.00 cass
$ 27.00 disk

CRIBBAGE for the VIC-20 or COMMODORE-64
Another fine game to run on your VIC-20 or CBM-64. This is the same classic card game that has been played for many years. Your computer does the dealing and scoring – quickly and accurately.
$ 24.00 cass
$ 27.00 disk

TINY BASIC COMPILER for the VIC-20, CBM-64 or PET/CBM
Here's a useful and educational tool for users. The TINY BASIC COMPILER brings you the high-speed advantages of a compiled language at a very modest cost. If you want to gain the advantages of a compiled language, learn more about the workings of compilers or delve into the VIC-20, CBM-64 or PET/CBM's firmware, then try our TINY BASIC COMPILER. For the VIC-20 with a *3K, 8K or 16K memory expander; for* PET/CBM *with OLD, NEW or 4.0 ROMS or* 8032; for the COMMODORE-64.
$ 24.00 cass
$ 27.00 disk

GRAPHVICS super graphics for the VIC-20
You must have heard about the High Resolution and Multicolour graphics that are built into each and every VIC. HiRes gives you 152 x 160 points and Multicolour 76 x 80 right on your VIC's screen. Now you can utilise both with the most versatile graphics package yet available for the VIC. GRAPHVICS gives you 2 screens, one for normal text and one for graphic displays. The function keys allow you to switch back and forth between the two screens. On the GRAPHVICS screen you have control over 24,000 individual points! And you can mix both HiRes and Multicolour modes on the same screen to create spectacular graphic pictures on a $299 computer.
$ 28.50 cass
$ 31.50 disk

VIGIL for the VIC-20 or PET/CBM
Includes nine games. VIGIL stands for Video Interactive Game Interpretive Language. Vigil is a simple, new language for writing game and graphics programmes for your VIC or PET micro. Vigil brings you more than 60 powerful commands for manipulating graphics figures in double density on your VIC or PET screen. Complete with manual. *It requires a 3K, 8K or 16K memory expander on the VIC 20.*
$ 33.00 cass
36.00 disk

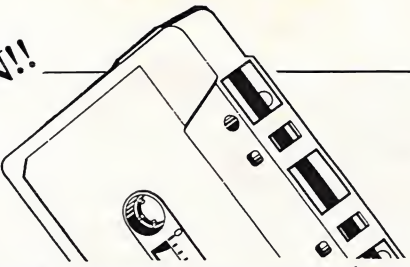
JOYSTICK PAINTER for the VIC-20
Here's one of the easiest ways yet to become an artist. VIC JOYSTICK PAINTER places a paintbrush in your hand. Use your joystick to draw pictures, graphs, figures.
$ 24.00 cass
$ 27.00 disk

I-CHING the Fortune Teller for the VIC-20
I-CHING is one of the oldest and most revered of fortune tellers. Now you can delve into your future with this colourful version of I-CHING based on ancient Oriental philosophy. The beauty of I-CHING is that you can go as far into it as feels comfortable. I-CHING comes as a complete package: the cassette program (or disk) with living colour and sound; the users introductory manual to get you started; and a 275+ page book to guide you through the hexagrams and Yin and Yang lines.
Requires either 8K or 16K expansion.
$ 28.50 cass
$ 31.50 disk

TINY PILOT
PILOT stands for Programmed Inquiry Learning or Teaching.
PILOT is an easy to learn language in which to program computer aided instruction lessons. Our version of TINY PILOT is so simple to use that you'll be writing your first program in minutes.
$ 24.00 cass
$ 27.00 disk

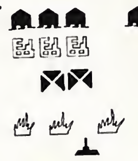
JOIN THE CLUB
★ Exchange cartridge service
★ Bonus discounts from the VIC CENTRE CATALOGUE
★ Club member specials

HEY LOOK AT THIS SELECTION!!

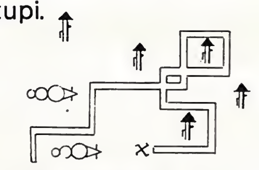


ROMIK SOFTWARE (UK)

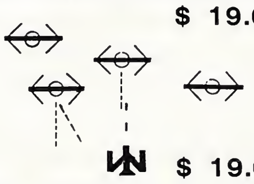
MARTIAN RAIDER – Skim as close as you dare to the surface of the planet, devastating the Martian cities, destroying ammunition dumps (gaining more time), shooting down the ground to air missiles and U.F.O's, dodging or blasting meteorites. **$ 19.00**



SEA INVASION – Fight off the attacking sea creatures as long as you can. Shoot the whale for a surprise score. Watch out for the crabs, starfish and octupi. **$ 19.00**

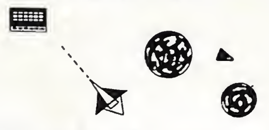


SHARK ATTACK – You are in shark infested waters – your only protection an atomic net which you trail behind you – protecting yourself and ensnaring sharks – careful of the octupi! **$ 19.00**



SPACE ATTACK – You are the pilot of an intergalactic battleship, fight your way through waves of alien fighters. **$ 19.00**

MIND TWISTERS – Four games to stretch your brain. **$ 19.00**



MOONS OF JUPITER – Send out your fighters from the mother ship to blast a way through the moons of Jupiter – look out for the U.F.O.'s and Grologs. *Requires 3, 8, 16 or Super Expander (3K)* **$ 19.00**

PIXEL (UK)

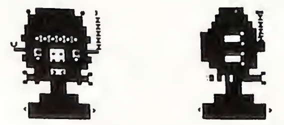
HARVESTER – unexpanded VIC – A cut throat strategy game to reap valuable Boosterspice around the planet Delta. HiRes graphics and lots of fun for two to four players. **$ 18.00**

BRAINSTORM – A battle of wits as you try to get three human explorers across a river of goo with three alien lifeforms that can literally blow your mind! **$ 18.00**

PIXEL POWER – To create user definable characters and save them with your own programmes. *8k expansion* **$ 18.00**

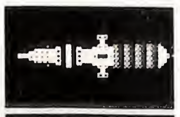
ENCOUNTER – Would you know what to do if you encountered extra-terrestial beings? In this exciting adventure, you are abducted by aliens and the space invaders play YOU! *16K expansion.* **$ 18.00**

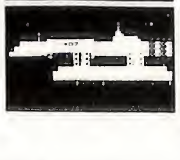
STARQUEST – A voyage of discovery and adventure in the cosmos. With the help of your onboard computer, you seek a habitable planet amidst the perils of deep space. *16K expansion.* **$ 18.00**



ZOR – Two mighty robots are designed for one purpose – to fight to the death. In the style of a mediaeval dual, you must do battle with the 'Champion of Zor' to save the planet Earth. *16K expansion.* **$ 18.00**

SUBSPACE STRIKER – It comes from out of nowhere and then vanishes back into the ether. With your deadly antimat torpedos, you unleash havoc in the Federation's spacelanes! *16K expansion.* **$ 18.00**





TRADER – Actual 3 x 16K parts to this program. It is hard enough to look at an amorphus hydrosilicon blob from PSI, never mind swing a deal with one, but when they ask to pick your brains. *16K expansion.* **$ 18.00**

MR MICRO LTD. (UK)

GREAT BALLOON RACE
Graphics Game for keyboard or joystick **$ 19.00**

MYSTERIOUS ISLAND
Recommended multi-tape adventure game for keyboard or joystick – needs *16K* **$ 19.00**

VIC VALUE NO 1
Keyboard or joystick – four games – Helicopter Lander, Dragon, Hunter, Alien Pilot **$ 19.00**

RAYFLECTION & MICROMIND
One game involves firing out rays to find the atoms, other is a variant of 'Mastermind'. **$ 19.00**

RABBIT SOFTWARE (UK)

RAID ON ISRAM (SCRAMBLE)
Raid on Isram is possibly the most challenging game ever introduced for the VIC-20. You must guide your craft through many perils to eventually get to your home base! **$ 17.00**

ORBIS
Defend your Uranium Fuel Dump from the invading "Zylons" by laying "space mines" in their path as they slip through the "energy field" but be careful as you only have a limited number of mines. **$ 17.00**

KRELL
Your mission is to defend the poor "Zymwatts" from the evil "Tharg" which sends "energy bolts" against their brick defences. You must destroy the "Tharg" before he kills all the "Zymwatts" which is not so easy as to get at the dreaded "Tharg" you must fend off his guardians. **$ 17.00**

UMI GAMES (USA)

SPIDERS OF MARS
Space battle. Protect your homeland against the Saturnian Bats and the Jovian Hornets. **$ 54.95** cart

METEOR RUN
Conquer planet Aldebaran, fight the aliens on the way and negotiate the deadly meteor fields. **$ 54.95** cart

AMOK
You are trapped in Amok, a deserted space station, try to find your way out but look out for the deadly robots. **$ 39.95 cart $ 16.00 cass**

CREATIVE SOFTWARE (USA)

coming soon

Serpentine	$ 55.00 cart
Astroblitz	$ 55.00 cart
Trashman	$ 55.00 cart
Terraguard	$ 55.00 cart
Videomania	$ 55.00 cart
Choplifter	$ 55.00 cart
Apple Panic	$ 55.00 cart

AUTOMATED SIMULATIONS (USA)

coming soon

Richochet	$ 30.00 cass
Datestones of Ryn	$ 30.00 cass
Invasion Orion	$ 36.00 cass
Sword at Fargoal	$ 42.00 cass
Rescue at Ryel	$ 42.00 cass
Crush, Crumble, Chomp	$ 42.00 cass
Monster Maze	$ 55.00 cart
Plattermania	$ 55.00 cart

HES SOFTWARE (USA)

Victrek	$ 24.00
Lazer Blitz	$ 24.00
Tank Wars	$ 24.00
Concentration	$ 24.00
Dam Bomber	$ 24.00
Fuel Pirates	$ 24.00
Pak Bomber	$ 24.00

LLAMASOFT (UK)

ABDUCTOR
A classic new space game! ZAP the swirling alien hordes before they ram you and abduct your humanoids.
IN MACHINE LANGUAGE. REQ. JOYSTICK **$17.00**

THE VIC CENTRE

In conjunction with CW Electronics
416 Logan Road.
Stones Corner Brisbane
4120 P.O. Box 274
Sunnybank Q'ld 4109.
Australia Tele: (07) 397 0808
397 0888 Telex: AA40811

Converting PET Programs for the Commodore 64

Many owners of the new Commodore 64 will have access to a large number of programs written originally for the PET computer. It is natural for these people to ask "What is involved in converting these programs so that they will run on the 64?". This article will attempt to detail some of the steps involved and hopefully make the conversion somewhat easier. I will only be discussing conversions involving 2.0 and 4.0 ROM PETs. Those interested in converting programs from 1.0 ROM PETs should be able to make the additional changes necessary.

In many cases, a PET program will run immediately on a 64. In some cases, a few minor changes will make the program workable. In a few cases, major surgery will be required, and in some instances, unless you are heavily into machine language, the conversion will be impossible. The type of conversion required will depend on the makeup of the original program.

As I said above, some programs will run immediately on the 64. These programs will be written entirely in BASIC and will not make use of the commands POKE, PEEK, WAIT, SYS, and USR. The easiest way to determine if a program falls in this category is to simply load the program into the 64 and run it. If it works, great. Otherwise read on.

Note: All BASIC programs for the PET will load into the 64 correctly. This may seem surprising since a PET program is stored in memory starting at location 1025 while 64 programs normally start at 2049. Such loads are successful because of a relocation feature incorporated into the Commodore 64 (and also the VIC 20) computer. These computers will automatically load a program at the START OF BASIC (wherever that happens to be), unless told to do otherwise (see your manual to see how to tell it to do otherwise).

I should point out that I have had some difficulty loading programs that were saved on a PET with 1.0 ROMS. Such programs do list but the first line is usually mangled. This can be fixed up by deleting that first line and retyping it or by a few simple POKEs.

The Simplest Conversion
Of the programs that do require conversion, the simplest to fix are the ones that do not use the SYS or USR commands. They may use the POKE, PEEK or WAIT commands, but these can usually be fixed up by changing an appropriate address and possibly a corresponding numeric value.

For example, POKE 59468,14 is a command frequently found in PET programs to convert the screen display to lower case. If this command is executed on the 64, nothing drastic will happen but lower case is definitely not displayed. The correct command on the 64 is POKE 53272,23. Thus part of the conversion process will be to locate all POKE 59468,14 statements in the program and change them to POKE 53272,23. Similarly, all POKE 59468,12 statements will have to be changed to POKE 53272,21 (This converts the screen display to upper case and graphics).

The majority of 'fixes' can be achieved in this manner.

i) Find the address on the PET that is causing a problem.

ii) Find the corresponding address on the 64.

& iii) Make all changes involving that address.

What is needed then is a list of addresses for the PET that can cause problems and a list of the corresponding addresses for the 64.

Actually, with a little more work, we can even do better. Ideally, a program should be able to run on any machine—PET with 2.0 ROMS, PET with 4.0 ROMS, and the Commodore 64.

This can be achieved for the upper/lower case conversion above in the following way.

Assume first that the program is running on a PET. Somehow have the computer execute the following commands:

```
3000 TEXT = 59468:  REM Address to be poked for
                        upper/lower case
3010 UC = 12:       REM Value to be poked for
                        upper case
3020 LC = 14:       REM Value to be poked for
                        lower case
```

On the other hand, if the program is running on a 64, have it execute the following:

```
3100 TEXT = 53272
3110 UC = 21
3120 LC = 23
```

Now change all POKE 59468,12 statements to POKE TEXT,UC and all POKE 59468,14 statements to POKE TEXT,LC. After these changes are made, the correct case will be displayed regardless of which computer the program is running on. If all other problem addresses can be fixed up in this manner, then we are well on our way to converting the program to work on all three computers.

Which Computer are You?

The first task then is to somehow identify what type of computer a program is running on.

There already is a standard technique for identifying whether a PET has 2.0 ROMS or 4.0 ROMS; namely,

```
110 IF PEEK (50003) = 160 THEN ... : REM 4.0
    ROMS
120 IF PEEK (50003) = 1 THEN ... : REM 2.0
    ROMS
```

PEEKing location 50003 on a 64 will usually yield a zero. I say 'usually' because 50003 is a RAM location on the 64 and is normally unused. However, machine language routines can be placed in that area and so you cannot be 100% sure what location 50003 will contain. The sequence below will set around this problem and will identify the type of computer correctly without destroying any machine code already there.

```
100 X = PEEK (50003): POKE 50003,0:Y = PEEK
    (50003)
110 IF Y = 160 THEN COMP$ = "4.0":REM 4.0
    ROMS
120 IF Y = 1 THEN COMP$ = "2.0":REM 2.0 ROMS
130 IF Y = 0 THEN POKE 50003,X:COMP$ = "64":
    REM COMMODORE 64
```

The statement POKE 50003,0 in line 100 has absolutely no effect on 2.0 PETS or 4.0 PETS since location 50003 is in ROM. On the 64 however, it puts a zero into that RAM location. Notice that the original value in location 50003 is saved by the statement X = PEEK(50003) and restored again in line 130 if the computer is identified as being a 64. Note the use of the variable COMP$ to identify the type of computer just in case it is needed again later in the program.

Now the conversion process should be clear. It should include the following:

1) At the beginning of the program, jump to a subroutine that identifies the type of computer that the program is currently running on.

2) In that subroutine, initialize a set of standard variables (such as TEXT, LC, UC, etc.) to the correct values for that computer.

3) Change all references to numerical addresses or values to the corresponding standard variables.

Here is a sample initialization routine.

```
10 GOSUB 60000
20 REM MAIN PROGRAM
60000 X = PEEK(50003):POKE 50003,0:Y =
      PEEK(50003)
60010 REM INITIALIZE VARIABLES COMMON
      TO 2.0 & 4.0 PETS
60020 TEXT = 59468:UC = 12:LC = 14:SCREEN =
      32768:HIV = 144
60030 NUMCHAR = 158:KEY = 151:NOKEY = 255
60040 IF Y < > 1 THEN 60100
60050 REM INITIALIZE VARIABLES PECULIAR
      TO 2.0 PETS
60060 COMP$ = "2.0":ENA = 46:DIS = 49
60070 RETURN
60100 IF Y < > 160 THEN 60200
60110 REM INITIALIZE VARIABLES PECULIAR
      TO 4.0 PETS
60120 COMP$ = "4.0":ENA = 85:DIS = 88
60130 RETURN
60200 IF Y < > 0 THEN 60300
60210 REM INITIALIZE VARIABLES PECULIAR
```

```
TO THE 64
60220 COMP$ = "64":TEXT = 53272:UC = 21:LC =
   23:SCREEN = 1024:HIV = 788
60230 NUMCHAR = 198:KEY = 203:NOKEY =
   64:ENA = 49:DIS = 52
60240 POKE 50003,X:RETURN
60300 PRINT "I DON'T RECOGNIZE THIS COM-
   PUTER.":END
```

The variables SCREEN, NUMCHAR, etc. will be explained shortly.

More Problem Areas

Upper/lower case conversion is certainly not the only problem area. Another potential one is the screen.

1. The Screen

On the PET the screen is found in memory locations 32768-33767. On the 64, it is found in locations 1024-2023.

If all output to the screen is obtained through the use of PRINT statements, then absolutely no problem will arise. If however, the output is POKEd to the screen, then changes will be required.

These changes are best achieved by assigning a value to the base address of the screen and then using an appropriate offset from that base.

For example the base address of the screen on the PET is 32768 while on the 64 it is 1024. Therefore, the first thing to do is to assign values to the standard variable SCREEN as follows:

```
SCREEN = 32768   if on a PET
SCREEN = 1024    if on a COMMODORE 64
```

i) Poking a single value onto the screen.

A statement of the form

```
POKE 32956,61
```

on a PET has to be changed as follows:

First, calculate the offset.

```
Offset = 32956-32768
       = 188
```

Then, change POKE 32956,61 to
POKE SCREEN + 188,61

The resulting statement will work on either a PET or a 64 (assuming SCREEN has been properly initialized).

Notice that the 61 does not have to be changed as these values are the same for both PETS and the 64

ii) Poking with a loop.

The following is a typical PET routine that POKES a border of 'reversed diamonds' around the screen.

```
100 FOR I = 32768 TO 32807 : POKE I, 218 : NEXT
110 FOR I = 32847 TO 33767 STEP 40 : POKE I,218 :
   NEXT
120 FOR I = 33766 TO 33328 STEP - 1 : POKE I,218
   : NEXT
130 FOR I = 33688 TO 32768 STEP - 40 : POKE
   I,218 : NEXT
```

This can be changed to work on both PETS and 64 by changing each screen address as above.

```
100 FOR I = SCREEN TO SCREEN + 39 : POKE
   I,218 : NEXT
110 FOR I = SCREEN + 79 TO SCREEN + 999 STEP
   40 : POKE I,218 : NEXT
120 FOR I = SCREEN + 998 TO SCREEN + 990 STEP
   - 1 : POKE I,218 : NEXT
130 FOR I = SCREEN + 920 TO SCREEN STEP - 40
   : POKE I,218 : NEXT
```

Or better yet

```
100 FOR I = 0 TO 39 : POKE SCREEN + I,218 :
   NEXT
110 FOR I = 1 TO 24 : POKE SCREEN + 39 +
   I*40,218 : NEXT
120 FOR I = 38 TO 0 STEP - 1 : POKE
   SCREEN + 960 + I,218 : NEXT
130 FOR I = 23 TO 1 STEP - 1 : POKE SCREEN +
   I*40,218 : NEXT
```

2. Clearing the Keyboard Buffer

The PET is able to retain up to ten keystrokes in a buffer, enabling you touch typists to type as fast as you can without losing any keystrokes. This can sometimes add extra unwanted characters to the beginning of an input, so a common technique in PET programming is to clear the keyboard buffer before each input is requested. This can be accomplished in a couple of ways.

```
100 FOR I = 1 TO 10 : GET A$ : NEXT
```

or

```
100 POKE 158,0
```

The first method will work as is on the 64. The second method must be changed.

On 2.0 and 4.0 PETS, location 158 always contains the number of characters in the keyboard buffer. On the 64 this value is stored in location 198. Thus if we assign values to the standard variable NUMCHAR as follows:

```
NUMCHAR = 158 if on a PET
NUMCHAR = 198 if on a 64
```

and change all references to POKE 158,0 to POKE NUMCHAR,0, then the resulting statement will work on both computers.

3. Pausing Until any Key is Pressed

Here again two techniques are commonly used.

```
100 GET A$ : IF A$ = "" THEN 100
```

is certainly the simplest and will work on both computers.

```
100 POKE 158,0 : WAIT 158,1 : POKE 158,0
```

is another technique and will have to be changed to

```
100 POKE NUMCHAR,0 : WAIT NUMCHAR,1 :
POKE NUMCHAR,0
```

4. Which Key is Pressed

A common technique used on the PET, especially in games, is to PEEK at location 151 to see if a key is being pressed and if so which one. Depending on which key is pressed a certain action is performed. This technique is frequently used in games that use the numeric keypad as a joystick. A sample sequence might be

```
500 X = PEEK(151)
510 IF X = 255 THEN 1000 : REM NO KEYPRESS
520 IF X = 18 THEN 2000 : REM 2 KEY IS PRESSED
530 IF X = 50 THEN 3000 : REM 8 KEY IS PRESSED
  etc.
```

The conversion here is a little more complicated but is still possible. First, we need to know that location 151 on the PET corresponds to location 203 on the 64. Then assign the following values to the standard variable KEY:

```
KEY = 151   if on a PET
KEY = 203   if on a 64
```

Replacing line 500 with

```
500 X = PEEK(KEY)
```

gives us a start with the conversion.

Another problem occurs with the values stored in location 151 (or 203) when a key is not being pressed. Location 151 on the PET contains 255 while location 203 on the 64 contains 64. This time we will use the standard variable NOKEY and initialize it as follows:

```
NOKEY = 255   if on a PET
NOKEY = 64    if on a 64
```

Line 510 is then replaced with

```
510 IF X = NOKEY THEN 1000
```

There are two problems associated with the other keys. First, location 151 will contain a certain value on the 2.0 machines, the same value on the Skinny 40 (i.e. the 9 inch screen) machines, but a different value on the Fat 40 machines. There is no standard way, that I am aware of, for distinguishing between a Skinny 40 and a Fat 40 machine. But PEEKing at location 57344 will do as well as any other. On a Skinny 40 you will get a value of 169 while on a Fat 40 you will get a value of 76.

The easiest way to see what value is stored in location 151 is to run the following program segment and press any key that you wish to test.

```
100 KEY = 151 : REM KEY = 203 ON THE 64
110 PRINT PEEK(KEY) : GOTO 110
```

The second problem arises from the fact that the 64 does not have a numeric keypad and using the numbers 2,4,6, and 8 to simulate a joystick is unacceptable. It would be much better to use the keys I,J,K, and M or some other suitably arranged set of keys.

My suggestion for getting around this is to first settle on the keys that you wish to use on each machine (they don't have to be the same). Find the values corresponding to these keys by running the short program above and store these values in corresponding standard variables which I like to designate K1, K2, K3, etc. (for KEY1, KEY2, KEY3, etc.) Then lines 520 and 530 can be replaced by the following:

```
520 IF X = K1 THEN 2000
530 IF X = K2 THEN 3000
  etc.
```

5. Disabling the Stop Key

The stop key on the PET can be disabled by altering the Hardware Interrupt Vector. For example.

```
POKE 144,49   for 2.0 PETS
POKE 144,88   for 4.0 PETS
```

will disable the stop key (and the time clock as well).

The corresponding command on the 64 is

```
POKE 788,52
```

To enable the stop key again

```
    POKE 144,46   for 2.0 PETS
    POKE 144,85   for 4.0 PETS
and POKE 788,49   for the 64
```

These can be replaced by

```
    POKE HIV,DIS   to disable
                   the stop
                   key
and POKE HIV,ENA   to enable
                   the stop
                   key
```

after appropriately initializing the variables HIV, DIS, and ENA. On the 64, the program can still be stopped by pressing the RUN/STOP and RESTORE keys simultaneously, but this will prevent stoppage of a program due to accidently pressing the STOP key.

A good question to ask is "How do you know what value is to be stored in these locations?". The PET program actually tells you the location to POKE as well as the value, but the value to be POKED on the 64 is usually different (c.f. disabling the stop key above or converting to upper/lower case). A memory map will tell you what location to POKE on the 64, but it will not tell you what value to POKE it with. A good start is to PEEK that location from direct mode and make note of the value. Do this for all three machines and it will tell you the 'normal' state of that location. For example, PEEKing location 144 on 2.0 PETS and 4.0 PETS yields 46 and 85 respectively. PEEKing at 788 on the 64 yields 49. Observing that the value to disable the stop key on the 2.0 & 4.0 PETS are each three more than the 'normal' value, a good start to finding the correct value on the 64 is to add 3 to the 'normal' value of 49, obtaining 52. This process will work for more than 90% of the problem values. It is that last 5–10% that makes the conversion challenging.

It would be impossible to list all problem locations and their 'fixes' here (I will list what I feel are the more common ones below). Instead I have attempted to give you a feeling for how the conversion should proceed. The proper tools that are required are the excellent memory maps (both zero page and ROM routines) published for all three computers in COMPUTE! by Jim Butterfield. Another excellent source is the book *Programming the PET/CBM* by Raeto Collin West and I am sure there are others.

Some of the More Common Problem Locations

2.0 PET	Location on 4.0 PET	64	Suggested Name	Description
40–41	40–41	43–44	SBAS	Start of BASIC text
42–43	42–43	45–46	SVAR	Start of variables
44–45	44–45	47–48	SARR	Start of arrays
46–47	46–47	49–50	EARR	End of arrays
52–53	52–53	55–56	TMEM	Top of memory
144–145	144–145	788–789	HIV	Hardware Interrupt Vector
151	151	203	KEY	Which key is pressed
158	158	198	NUMCHAR	Number of characters in keyboard buffer
159	159	199	RVS	Screen reverse flag
167	167	204	CRSR	Flag for flashing cursor in GET statements
196	196	209	SLO	Pointer to screen line (low/high format)
197	197	210	SHI	
198	198	211	CH	Horizontal position of cursor
216	216	214	CV	Vertical position of curosr
623	623	631	BUFF	Start of keyboard buffer
634	634	—	—	Start of first cassette buffer
826	826	828	CAS	Start of second cassette buffer
32768	32768	1024	SCREEN	Start of screen memory
59468	59468	53272	TEXT	Poke location for upper/lower case
64721	64790	64738	—	Simulates power on reset
			Special Values	
12	12	21	UC	Upper case
14	14	23	LC	Lower case
255	255	64	NOKEY	No key is pressed
46	85	49	ENA	Enable stop key
49	88	52	DIS	Disable stop key

A Couple of Cautions

1. The PET and the 64 only recognize the first two letters of a variable name. When converting a program you must make certain that the variables already present in the program do not conflict with the standard variables suggested above. If there is a conflict, change whichever you feel is easier.
2. If a program is to be used both on a PET and a 64, then the changes should be made on and saved with a PET computer. The reason for this is that a program saved on a 64 will not load properly on a PET, due to the lack of a relocation feature in the PET (But see below).

   If the changes are made on a PET, then a utility such as BASIC AID or POWER will be invaluable since you can type in such things as

   **FIND \POKE **

   and all lines that contain a POKE statement will be listed, making it easier for you to make the necessary changes and to make certain that you have found all of them.

   Similarly you can type in

   **FIND \SC **

   to see if there are any variables in the program that will conflict with the standard variable SCREEN.

Programs That Contain SYS or USR Commands

These programs will require that you be somewhat familiar with machine language in order for you to be able to make the necessary conversions. Such changes are beyond the scope of this article. However, let me say that these M/L routines themselves fall into a number of categories.

1) The routine works on the 64 as is. (Few routines will likely fall in this category)

2) The routine will work with a simple address change. (These are frequently ROM routines such as the routine for resetting the entire stack)

3) The routine will work with some minor changes. An example here could be a routine to reverse a portion of the screen. Chances are the only changes necessary would be for the location that determines the base address of the screen. However, if parameters are passed in the calling statement, then the location of certain ROM routines (such as checking for a comma) might have to be changed as well.

4) The routine will require major surgery before it will work. A program like BASIC AID or MICROMON would fall into this category. Such programs should be left to the experienced users.

Other areas that may require major changes are those programs that make use of the BASIC 4.0 disk commands. Some of these can be fixed up easily, but some are extremely difficult (e.g., those that make use of Relative Record files).

Programs that make use of CB2 sound will still run on the 64, but no sound will be produced. The POKEs that the PET uses to produce these sounds will POKE into the ROMS of the 64 and hence do no harm. Once you are familiar with the sound process on the 64, here is a good place to make use of the variable COMP$. For example, suppose lines 1000–1030 in the PET program are used to produce the sound. Leave these lines exactly as they are and add a similar routine for producing sound on the 64 beginning with

1031 IF COMP$ < > "64" THEN 1040

Your sound routine can be placed in lines 1032–1039 and the rest of the program should proceed as normal.

Loading Programs Saved on the 64 Into the PET

As mentioned above, programs saved on the 64 do not load properly into a PET. It is not a difficult procedure to correct this shortcoming however. Here are the steps.

1. Type in a dummy line 0 into the PET.

   0 REM

   will do.
2. Type in **POKE 2048,0**
3. LOAD in the program that was saved on the 64 as you normally would.
4. Type in **POKE 1025,1 : POKE 1026,8**

You should now be able to LIST the program including the dummy line 0 that you typed in initially. Delete this line by typing in 0 <RETURN>. The process is now complete. You can save the program to cassette or disk. The next time that you load it into your PET, it will load normally. If you are using a disk, you will notice that the program is 3 blocks longer than the original even though it is the same program. The reason for this is that the Start of Variables pointer did not get changed properly. An experienced programmer can set into the monitor and make the necessary changes without too much difficulty, but the program will operate correctly without making this change. C=

Your First Practical Program

by Michael S. Tomczyk

Reprinted from Commodore Magazine, April, 1982

Creating "Practical" Programs

Some people can't cook (like the VIC Magician). But by following a recipe point by point, even the VIC Magician can come up with some tasty concoctions. Programming your VIC 20 is just like cooking—the ingredients in a program can be mixed and matched like a recipe. All you need is a menu of neat little programs you can put together in various ways to create sophisticated practical programs for home, school or business. This modular approach helps you do complex computing with a minimum of experience and allows you to take advantage of some sophisticated programming techniques, even if you're just a beginner.

One Step at a Time

Writing a practical program is easy. Just take everything one step at a time. You'll find that most practical programs use the 6 steps shown below. There are many other approaches but this should give you a good start. Oh. . .one caution. . . don't get bogged down with "cosmetics" like graphics or screen placement. Your first task is to get the program WORKING in terms of instructions, calculations and results. The cosmetics and adjustments are the LAST elements to add to your program. To use our cooking analogy, broil your steak first, then add the seasoning. Here then are 6 elements involved in most practical programs:

Illustration by Don Dyen

1. DISPLAY (PRINT) QUESTIONS OR CATEGORIES ON THE SCREEN.
2. CONVERT THE USER'S TYPED-IN RESPONSES TO "INPUT VARIABLES." (In other words we translate what the user types in from the keyboard into

short string or numeric variables the VIC can understand and work with.)
3. USE THE INPUT VARIABLES TO CALCULATE THE RESULTS YOU WANT TO SHOW.
4. DISPLAY (PRINT) THE RESULTS OF THE CALCULATIONS, ALONG WITH SOME DESCRIPTIVE WORDS.
5. REPEAT THE PROGRAM OR END THE PROGRAM.
6. ADD COSMETICS (to make the program "friendly").

Each program is different, of course, but these are the basic parts of most practical programs. The hard part is putting it all together. Most of us are intimidated by long programs with lots of fancy commands and mysterious variables, but those programs didn't start out to be that long or fancy. They started very simply and grew as the programmer kept adding elements to make the program more powerful or "friendly."

Plan Your Program Carefully

So you're ready to write your practical program. The first thing to do is make a CHECKLIST of what you want your program to accomplish. Try to write it in chronological order. It it's a long list, break the elements down into the simplest steps and build the program slowly, starting with the program's LOGIC. Here's an example of the checklist used for an INCOME/EXPENSE program described in detail below:

1. NAME OF PROGRAM: INCOME/EXPENSE BUDGET
2. CLEAR SCREEN
3. ENTER TOTAL INCOME
4. ENTER EXPENSE CATEGORY (3 ITEMS)
5. ENTER EXPENSE AMOUNT
6. ADD TOTAL EXPENSES (E1+E2+E3 = EXPENSES)
7. COMPUTE NET INCOME (INCOME—TOTAL EXPENSES = BALANCE)
8. FIGURE PERCENT OF TOTAL EXPENSES FOR EACH CATEGORY (E1/E = %)
9. EXPENSES = WHAT PERCENT OF INCOME? (EXPENSES/INCOME = %)
10. INCLUDE ROUNDING FUNCTION TO ROUND ALL NUMBERS TO 2 DECIMAL PLACES (CENTS) (POSSIBLE DEF FN).

From here, you can either do a more detailed programming flowchart using standard flowcharting symbols and notation, or you can simply make programming notes next to your original checklist, as shown above in parentheses.

Some Practical Programming Tips

Before we get into our program example, let's go through some fairly standard practical programming tips, with some shorter examples to show you how various techniques work.

CLEAR THE SCREEN

The first line in most practical programs is a clear screen command. Clearing the screen is covered in the VIC 20 user's guide, but here's a quick refresher. Normally, you CLEAR the screen by holding down the SHIFT key and hitting the CLR/HOME key. In your program, you will simply add an opening line number and PRINT the CLEAR command, like this:

```
10 PRINT" ♥ " (Type SHIFT CLR/HOME
              here. . .the reverse heart
              symbol appears whenever
              you CLEAR the screen
              inside quotation marks)
```

The CLEAR command doesn't have to be on a separate line. You can include it in your first PRINT statement, if you like, for example:

```
10 PRINT " ♥ MONTHLY INCOME"
```

USE PRINT STATEMENT TO PROMPT (INSTRUCT) USER

A "prompt" can be a question, word, phrase, number, instruction or category. It simply helps the user type in the right information. Here's an example of a PRINT statement that "prompts" the user to type in his monthly income:

```
10 PRINT "MONTHLY INCOME:"
```

USE INPUT & VARIABLES TO ACCEPT USER INFORMATION

An INPUT statement automatically places a question mark (?) on the next line after the prompt you PRINTed, and causes the VIC 20 to wait patiently until the user types in a response to the prompt. Here's an example of an INPUT statement:

YOU TYPE THIS:	VIC SCREEN SHOWS THIS:	
10 PRINT "MONTHLY INCOME:"	MONTHLY INCOME:	VIC displays this and "waits"
20 INPUT M	?	
30 PRINT "YOUR INCOME IS" M	1000 INCOME	← You enter this
(type RUN & hit RETURN)	YOUR INCOME IS 1000	VIC responds

What we did here was first PRINT the prompt message. . . then we told the VIC to wait for an INPUT (response) to be typed from the keyboard, and assigned the variable name M to the value representing monthly income. The PRINT message in line 30 simply proves that the VIC accepted this information.

INPUTs can and usually are on the SAME LINE as the PRINT statement. You can save memory space and increase your program efficiency by combining the PRINT and INPUT statements on one line. Be sure to separate both commands with a colon (:) as shown:

```
10 PRINT "MONTHLY INCOME":INPUT M
```

You can also combine several INPUTS on one line. For example:

```
10 PRINT "ENTER 3
   NUMBERS":INPUTA:INPUTB:INPUTC
20 PRINT A;B;C
```

VICtip for First-Time Computer Owners

TO ENTER A PROGRAM LINE INTO THE VIC 20—just type the line number, the program commands or statements, and hit the RETURN key.

TO SEE THE PROGRAM LINE(S) OPERATE—type the word "RUN" and hit RETURN.

TO SEE YOUR PROGRAM DISPLAYED LINE-BY-LINE—for editing purposes, type the word "LIST" and hit the RETURN key.

TO CHANGE A PROGRAM LINE—either use the CURSOR keys to move to the line and change it, then hit RETURN, or retype the line number and hit RETURN (the new line with the same number will replace the old one with that number).

TO DELETE A PROGRAM LINE—simply type the line number of the line you want to delete and hit RETURN (to delete line 10 just type the number 10 and hit RETURN).

How Variables Work With INPUT Statements

Variables allow the VIC to accept information typed in by the user and then use that information in the program! Variables are in fact the key to interactive programming (this series places a lot of emphasis on variables because they're so important. . .your ability to program increases greatly if you understand how variables work).

You can think of a variable as a "storage compartment" where the VIC stores the user's response to your prompt question. For example, you can write a program that asks the user to type in his name. In this case, you might assign the variable **N$** to the name typed in. Now every time you PRINT N$ in your program, the VIC will automatically PRINT the name the user typed-in! Type the word NEW, hit RETURN, and try this example:

```
10 PRINT"YOUR NAME":INPUT N$
20 PRINT"HELLO,"N$
```

You might have noticed that we used variables like A,B,C and M to represent numbers in earlier examples, but here we use the variable N$ to represent a name. We used N to remind ourselves that this variable stands for "NAME" and we used a dollar sign to signify a string variable. This is important because variables come in two flavors: numeric variables and string variables.

NUMERIC VARIABLES are used to store number values such as 1, 100, 4000, etc. A numeric variable can be a single letter (A), any two letters (AB), a letter and a number (A1), or two letters and a number (AB1). You can save a little memory by using shorter variables but letters and numbers (A1, A2, A3) are often best if working with different categories in the same program. There is also a special type of numeric variable called an integer variable which eliminates decimal places from your numbers (helpful if you want calculations represented as whole numbers instead of numbers with long decimals). To get integer (whole) numbers, simply put a percent (%) sign at the end of your variable name (i.e., A1% or AB%) which makes the VIC treat all numbers typed in as whole numbers only, dropping any decimal places. Note that the percent sign here doesn't mean you are calculating real percentages. It's simply a symbol used with integer type variables.

STRING VARIABLES look just like numeric variables except they end with a dollar sign ($) like A$ or A1$. String variables are used to store words, phrases, sentences, graphic symbols and "numbers which are used like words."

By "numbers used like words" we mean numbers that will not be used in a calculation. Your social security number is an example of a number that is identified with a string variable instead of a numeric variable, because your SSN is used like a "label" instead of a "value."

Another good example is your **age**. If you ask the user to type his age for "reference" in a program, you might use a string variable because you will not be doing any calculation using his age. . .BUT. . .if you plan to calculate his date of birth by subtracting his age from this year, you are using age as a **number value** and must use a numeric variable. String variables take the same form as numeric variables, except they are followed by the **S-shaped dollar sign** ($). Examples of string variables are: A$, AB$, A1$, AB2$, and so on. Here are some examples:

Example 1

```
10 PRINT"ENTER A NUMBER":INPUTA
20 PRINTA
```

Example 2

```
10 PRINT"ENTER A WORD":INPUT A$
20 PRINTA$
```

Example 3

```
10 PRINT"ENTER A NUMBER":INPUT A
20 PRINT A"TIMES 5 EQUALS"A*5
```

Note in Example 3 that the MESSAGES or PROMPTS are INSIDE the quotation marks, while the variables are OUTSIDE. In line 20, you PRINTed A (the number you entered when prompted), then the message "TIMES 5 EQUALS," and then a CALCULATION (multiply the number A*5).

Calculations Make Practical Programs "Practical"

Calculations are important in most programs, from games to business aids. You have a choice of using either actual numbers or variables, when doing calculations inside a program, but if you're working with numbers supplied by the user you must use numeric variables. Begin by asking the user to type in two numbers, like this:

```
10 PRINT"TYPE 2 NUMBERS":INPUTA:INPUTB
```

Now multiply those numbers together to create a new variable C (C is number A multiplied by number B). The calculation uses the variables like this:

```
20 C=A*B
```

. . .and this lets you now PRINT the result as a MESSAGE.

```
30 PRINTA"TIMES"B"EQUALS"C
```

Enter these 3 lines and RUN the program. Note again that numeric or string variables are always OUTSIDE the quotation marks but any message information must be INSIDE the quotes. That's why we put words like TIMES and EQUALS inside quotation marks. Similarly, if we want a dollar sign in front of a number we've calculated, the dollar sign ($) must be PRINTed INSIDE quotes, in front of the numeric variable, like this (hit RUN/STOP and RESTORE, then enter this line and hit RETURN, then type RUN and hit RETURN):

```
40 PRINT"$"C
```

The dollar sign goes in quotes because the variable C only represents a number and can't contain the dollar sign. If the number represented by C was 100 then the VIC screen will display $100. However, if you tried to PRINT $C without putting the dollar sign in quotes, you'll get a SYNTAX ERROR IN LINE 40 error message because the dollar sign has to be inside the quotes.

One last tip about dollar signs. If you like, you can, in the first or second line of your program, create a variable that you can substitue for the dollar sign for use with numeric variables. If, for example, you type:

```
10 Z$="$"
```

. . .you can use the string variable Z$ wherever you want a dollar sign. Try this:

```
10 Z$="$":INPUTA
20 PRINTZ$A
```

The key is the Z$A in line 10. We begin by defining the dollar sign as a string variable called Z$. Then we INPUT a number which we call A. Finally, we PRINT both variables side by side and we get Z$ which is the dollar sign, and our number A printed together on the screen. By doing this, you can ALWAYS use Z$ (or whatever string variable you choose) next to a numeric variable when you need a dollar sign. . . and you'll probably find it easier than typing "$" every time you want to calculate dollars.

Time Delay Loops

Time delay loops are very helpful in slowing down a program, where appropriate. A full time delay loop looks like this: FORT=1TO 1000:NEXT and can be inserted almost anywhere in your program, either on a separate program line or inside a line, separated by colons. Change the duration of the delay by changing the "1000" to a larger number (slows down delay) or smaller number (speeds up).

A Word About Cosmetics

COSMETICS include all the small extras you can add to your program after the logic is working. Here's a short list of "extras" to consider as you polish and refine your program. . .but don't worry about these cosmetics until the LOGIC of the program is finished and working!

- Insert blank lines where appropriate to make your screen display more readable (a PRINT statement on a line by itself automatically inserts a blank line when the program is RUN).
- Consider different colors. . .check your user's guide to see how POKE36879,X works and consult the SCREEN AND BORDER COLOR appendix for color variations. Use this POKE as the first or second line in your program to make the opening display appear in special colors.
- Sound effects can jazz up your program. A good one-line sound effect format is described below.
- Title your program with a REVERSE title at the top. . .or . . .use the special title technique described below.
- Use upper and lower case letters instead of all capitals. Just type PRINTCHR$(14) in your program to switch to upper/lower case and type PRINTCHR$(142) to switch back to upper case only. These PRINTCHR$ statements should be used like this: 10 PRINTCHR$(14)"Income"
- Graphics are easy on the VIC 20, which includes a full set of "business" graphics to give you different types of lines and bars that allow you to separate different items by lines or bars, add highlights, etc.

A Screen Title Format

If you replace line 5 in the program example, you can include a screen title that looks like the title is part of the border.

```
5 POKE36879,30:PRINT "♥"
CHR$(18)"INCOME/EXPENSE PROGRAM"
```

We used line 5 because that makes it the first line in our program—the important thing is to make the screen format program the first line in the program (so the title comes on first). Note that a clear screen command will erase this title so if you clear the screen anywhere later in your program, you'll have to put this line in again, immediately after you PRINT the clear screen.

This is an illusion that actually comes from REVERSING the top line in the screen area to make it appear to be part of the border. The program line does the following things, in order:

1. POKEs the border to dark blue.
2. CLEARS the screen.
3. Turns on REVERSE lettering (hold down CTRL and type RVS ON key).

The result is a title in white letters across the top of the screen . . .actually the letters are inside the screen area, but because we REVERSED the top line, it looks like part of the border.

INCOME/EXPENSE BUDGET EXAMPLE

```
5   PRINT" ♥ "                                            (clears the screen)
10  PRINT"MONTHLY INCOME":INPUT IN                        (PRINT/INPUT statement)
20  PRINT                                                 (Inserts blank line)
30  PRINT"EXPENSE CATEGORY 1":INPUT E1$                   (Expense Cat 1=E1$)
40  PRINT"EXPENSE AMOUNT":INPUT E1                        (Expense Amt = E1)
50  PRINT                                                 (Blank line)
60  PRINT"EXPENSE CATEGORY2":INPUT E2$                    (Expense Cat 2 = E2$)
70  PRINT"EXPENSE AMOUNT":INPUT E2                        (Expense Amt 2 = E2)
80  PRINT                                                 (Blank line)
90  PRINT"EXPENSE CATEGORY 3":INPUT E3$                   (Expense Cat 3 = E3$)
100 PRINT"EXPENSE AMOUNT":INPUT E3                        (Expense Amt 3 = E3)
110 PRINT"♥"                                              (clear screen)
120 E=E1+E2+E3                                            (Add Expense Amts = E)
130 EP=E/IN                                               (Calculate Expense/Income%)
140 PRINT"MONTHLY INCOME: $"IN                            (Display Income)
150 PRINT"TOTAL EXPENSES: $"E                             (Display Total Expenses)
160 PRINT"BALANCE EQUALS: $"IN-E                          (Display Income—Expenses)
170 PRINT                                                 (Blank line)
180 PRINTE1$"="(E1/E)*100"% OF TOTAL EXPENSES"            (Lines 180-200 calculate
190 PRINTE2$"="(E2/E)*100"% OF TOTAL EXPENSES"            % each expense amount is
200 PRINTE3$"="(E3/E)*100"% OF TOTAL EXPENSES"            of total expenses)
210 PRINT                                                 (Blank line)
220 PRINT"YOUR EXPENSES="EP*100"% OF YOUR TOTAL INCOME"   (Display E/I%)
230 FOR X=1 TO 5000:NEXT:PRINT                            (Time Delay Loop)
240 PRINT"REPEAT? (Y OR N)":INPUTY$:IFY$="Y"THEN 5:IFY$="N" THEN GO TO 250
250 PRINT"♥ ":END
```

Changing Screen/Border Colors

Try adding the following line to your program:

5 POKE36879,X:PRINT " ♥"

where X can be any number between 1 and 255.

Adding Sound Effects

To use a sound effect, it's usually a good idea to include this line at the beginning of your program (it would normally start at line 10 as the first program line but we're using line 2 to add it to the beginning of our program above):

2 POKE36878,15:S1=36874:S2=36875:
S3=36876:S4=36877

Now you can use any of the VIC's four speakers by referring to S1, S2, S3 and S4. S1 is the deepest (lowest) voice and S3 is the highest (soprano). S4 is used for "white noise sound effects." Here is a sound effect you can add to your program above. Simply insert this line wherever you want a "beep" in your program (for example at Line 35):

35 POKES3,200:FORT=1TO200:NEXT=
POKES3,0

PROGRAMMER'S TIPS

The VIC Magician
Peeking Keys To Control Actions

By Michael S. Tomczyk
Product Marketing Manager

Here we are again . . . with more lessons to help you write your own games. As we've said before, many of the best gamewriting techniques aren't explained anywhere.

In this article we're going to explore several aspects of gamewriting that are hard to pick up if you're just getting started on your own—starting with a summary of how to make symbols appear AT RANDOM in various places on the screen, how to make them MOVE, and finally, how to "PEEK" ahead to establish "collision detection."

It will help a lot if you have your *VIC-20 Programmer's Reference Guide* or VIC owner's guide handy . . . and if you missed Part I and Part II in this gamewriting series, take a look at the review material on page 52 before going on.

Making Objects Appear At Random On The Screen
In Part I and II, we touched on how to make objects appear at random but now it's time to summarize that information. It never hurts to review how random numbers work, and the chart below will give you a quick reference when you want to use this "random" technique in your action games.

Most action games use the VIC's built-in random number generator to make objects appear haphazardly along one of the four screen borders (top, bottom, left and right), and then move across the screen. You can also make objects appear at random anywhere on the ENTIRE SCREEN, or in a particular row or column.

So you can better see what we're doing, let's use the variable TP to stand for the TOP ROW LOCATION on the screen, BT for the BOTTOM ROW LOCATION, LF for the LEFT COLUMN LOCATION, RT for the RIGHT COLUMN LOCATION and A for ANYWHERE ON THE ENTIRE SCREEN. **It's important to remember that we're talking about SCREEN LOCATION of a symbol. Study the following chart before reading on:**

FORMULA	HOW TO USE IN A GAME
TP=(22*RND(1))+7680	**TOP ROW:** Using TP as a screen location puts a symbol at a random position on the TOP ROW of the screen.
BT=(22*RND(1))+8164	**BOTTOM ROW:** Using BT as a screen location puts a symbol at a random position on the BOTTOM ROW of the screen.

L=INT(22*RND(1))+0
LF=7680—(22*L)
	LEFT COLUMN: Using LF as a screen location puts a symbol at a random position along the LEFT SIDE of the screen. The first "L" formula picks a number from 1 to 23 which is needed in the LF formula.

R=INT(22*RND(1))+0
RT=7701—(22*R)
	RIGHT COLUMN: Using RT as a screen location puts a symbol at a random position along the RIGHT SIDE of the screen. The first "R" formula picks a number from 1 to 23 which is needed in the RT formula.

A=(506*RND(1))+7680	**ANYWHERE ON THE SCREEN:** Using A as a screen location puts a symbol at a random position anywhere on the entire screen (note there are 506 locations on the VIC screen).

Let's see how these formulas work in a BASIC program. Type the word NEW and press return to erase any previous programs that might be in your computer, then enter this program exactly as shown:

```
10 PRINT" SHIFT CLR/HOME RANDOM SYMBOLS"
       (press RETURN)
20 TP=(22*RND(1))+7680
       (press RETURN)
30 BT=(22*RND(1))+8164
       (press RETURN)
40 L=INT(22*RND(1))+0:LF=7680+(22*L)
       (press RETURN)
50 R=INT(22*RND(1))+0:RT=7701+(22*R)
       (press RETURN)
60 A=(506*RND(1))+7680
       (press RETURN)
70 TC=TP+30720:POKETP,90:POKETC,4:GOTO20
       (press RETURN)
```

Whew! Lots of typing, huh? Well, we'll do a lot of things with this program so you won't have to keep retyping all those tricky formulas. (The VIC MAGICIAN doesn't like retyping formulas any more than you do!)

Type the word RUN and press the RETURN key to see the program work.

What happens? A series of red diamonds are appearing along the top of your screen! That's because in Line 70, we used the TOP ROW FORMULA we set up in Line 20 to put a random series of symbols on the screen. We got the POKE NUMBER (90) for the diamond symbol from the SCREEN CODE chart on Page 268 of your *Programmer's Reference Guide*, page 142 of your VIC owner's guide. Press the RUN/STOP key to stop your program and type LIST and hit RETURN to see the listing. Here's a line-by-line explanation of the program:

LINE 10 clears the screen and displays a title.

LINE 20 sets up a random formula for the TOP ROW screen location, which we've designated TP. You can use your own variables instead of these, incidentally (for instance, T, T1, TR or any other variable works just fine).

LINE 30 sets up a random formula for the BOTTOM ROW.

LINE 40 sets up a random formula for the LEFT COLUMN.

LINE 50 sets up a random formula for the RIGHT COLUMN.

LINE 60 sets up a random formula for ANYWHERE ON THE SCREEN.

LINE 70 starts out by establishing the COLOR LOCATION which matches the TOP ROW LOCATION. The color location of any symbol we are POKEing on the screen is always the number 30720 added to the screen location. So the color location of location TP must be TP + 30720. We will call the color location TC (for "TOP ROW COLOR") and get back to this in a moment.

Remember, you're using TP as a SCREEN LOCATION for the symbol you want to display AT RANDOM in the TOP ROW. We used the diamond symbol whose POKE NUMBER is 90 (from the SCREEN CODE chart in your manual), so we POKE TP,90 to display the symbol on the screen . . . but wait! To display a symbol you always have to make TWO POKES . . . the screen location AND the matching color location. So we'll POKE TC (the name we gave the color location for the top row) with the COLOR NUMBER 4 which is red (color numbers are always one less than the numbers shown on the keyboard, when POKEing colors). Finally, we GOTO Line 20 which sends the program back to get a NEW RANDOM NUMBER for the top row . . . and by the time the program comes back to Line 70, the variable TP has become a new random number . . . so the color location (TP+30720) also changes . . . and everything keeps going.

Confused? If so, take a look at the review on page 52 for more information on how Line 70 works. Once again . . . we defined the color location by adding 30720 to the screen location, which in this program is a RANDOM location called TP. We then POKE the screen location (TP) with the screen code number of the SYMBOL we want to display (in this case, a diamond whose screen code number is 90). Then we POKE the matching color location (TC) with a COLOR NUMBER, which is number 4 which is red. Try changing the color and symbol numbers (4 and 90) to get used to working with various settings.

Okay, you've seen the top row . . . how about the others? Well, all the formulas are still there in your program, although we only used one of them for this example. To use any of the other formulas, simply change Line 70.

To change a line, all you have to do is retype the entire line, including the line number, and press the RETURN key. So . . . retype Line 70 as follows to see the BOTTOM ROW example:

```
70 BC=BT+30720:POKEBT, 90:POKEBC,4:GOTO20
    (Press RETURN)
```

Type RUN and press RETURN. Notice you can still GOTO 20 in Line 70, because when the program goes there, it will "fall through" to Line 30 where the RANDOM BOTTOM ROW LOCATION is being set. Press the RUN/STOP key, then retype Line 70 as follows:

```
70 LC=LF+30720:POKELF, 90:POKELC,4:GOTO20
    (Press RETURN)
```

Type RUN and press RETURN. To try the next example, press the RUN/STOP key and retype Line 70 again as shown, then type RUN to see it work:

```
70 RC=RT+30720:POKERT, 90:POKERC,4:GOTO20
    (Press RETURN)
```

And . . . to see the "anywhere" example, try this Line 70:

```
70 AC=A+30720:POKEA,90:POKEAC,4:GOTO20
```

Adding a Bit of Music

You can add music to the examples we've just explored. Here's how: just add the "music lines" at Lines 80 and 85 and move the "GOTO" at the end of Line 70 to Line 90, as shown:

```
70 AC=A+30720:POKEA,90:POKEAC,4
    (press RETURN)
80 POKE36878,15:POKE36876,
    (100*RND(1))+150
85 FORT=1TO200:NEXT:POKE36876,0
90 GOTO20
```

Careful! This one will drive you bonkers if you listen to it long enough! In any event, here's how you did it:

LINE 70 is the same "ANYWHERE" example we used last. You simply deleted the GOTO 20 from the end of the line.

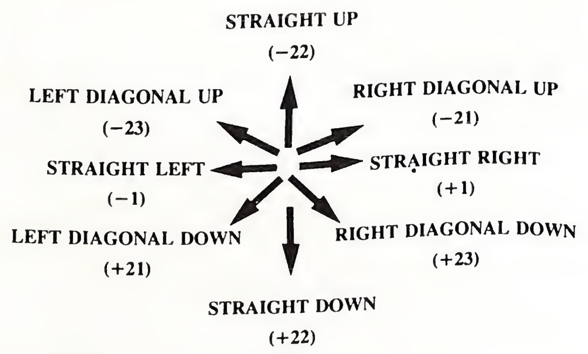
LINE 80 was automatically added to your program when you typed it in. POKE36878,15 sets the VOLUME at its highest level. POKE36875 is the POKE for "speaker 2" of the VIC's four internal "speakers." Usually, you POKE a speaker setting with a number from 128 to 255 to generate a musical tone. Here, instead of using one of those numbers, we used a RANDOM NUMBER to represent the musical tone. The RANGE of random musical settings is from 150 to 250. The 150 is the STARTING POINT and the 100 is the RANGE (150+100 = 250). The VIC will choose a different number from 150 to 250 each time it goes around and selects another diamond and plays the tone. The FOR...NEXT loop provides the DURATION the note plays, before we turn it OFF by POKEing36876,0.

LINE 90 is where we put the GOTO. We basically inserted the musical note being played in Line 80 after Line 70, but before the GOTO because we want to play the note before we go back around and display another diamond (and play another random note).

Are you beginning to get comfortable with random numbers? Just remember that the FIRST NUMBER on the left side of the formula is always the RANGE and the last number on the far right side of the formula is the STARTING POINT. So, for example, (506*RND(1))+7680 means we are selecting a random number starting at number 7680 with a range of 506, which means that the random numbers will be chosen from the set of numbers which includes 7680 to 7680+506, or 7680 to 8186.

Moving Symbols

To move a symbol on the screen, simply change the SCREEN LOCATION (the color location should automatically change because it is always the screen location plus 30720). Here, briefly, is how much you have to add or subtract to a screen location to move a symbol in one of the 8 possible directions:



Often, you may have a symbol appear at random on the left edge of the screen, and "fly" across the screen to the right. This is an easy movement because you simply add 1 to the screen location. You can add 1 to the location even if the location is "randomly" generated. Let's try an example:

We're going to retype our sample program a little, starting with a new Line 65. .STOP and LIST your program, then retype Lines 65-90 as shown:

```
65 E=LF+22
70 LC=LF+30720:POKELF−1,32:POKELF,90
    :POKELC,4
80 POKE36878,15:POKE36876,160:FORT=
    1TO10:NEXT:POKE36876,0
```

```
 90 LF=LF+1:LC=LC+1:IFLF=ETHENGOTO20
100 GOTO70
```

Type the word RUN and press RETURN to see the program work, then press the RUN/STOP key, type the word LIST and press RETURN.
Look complicated? It's not! Did you ever read a *Commodore* magazine and get boggled trying to understand all the variables like LF, RT, A, LC, etc.?? Pretty confusing, huh? You may not have realized it yet, but you have just written one of those "complicated" programs yourself! And you understand what all the variables mean (at least you can check back to the chart if you forget any). Well, now we're going to start USING some of those variables to make your diamond symbol move, and do some other interesting things as well. First let's look at these extra program lines:

LINE 65 is a special line which defines a NEW SCREEN POSITION CALLED "E" (for "END" of the line). E at this point in the program becomes the SCREEN LOCATION AT THE END OF THE LINE where the random LF begins. For example, if LF starts out at position 8010, then E is defined as 8010+22. or 8032 which is the beginning of the next line. In a little while, we'll use E as an "end of line limiter."

LINE 70 is the same as the LEFT COLUMN example we used earlier, with the exception that we added: POKELF-1,32. This ERASES the PREVIOUS SYMBOL. In other words, we are POKEing one symbol in the first (left) column, then POKEing the next symbol in the next position (LF+1). If we don't erase the previous symbol, we will wind up with a line of symbols stretching across the screen and that's not what we want because we're trying to create the illusion of movement, which is what animation really is. So we must ERASE LF−1 by POKEing a SPACE SYMBOL (number 32) into the previous location. That's what this POKE does.

LINE 80 is our music program except we've speeded it up by changing the 200 in the FOR...NEXT loop to 10. You can speed up or slow down the rate at which the diamonds move (and the duration of the musical tone) by changing this number. We also changed the tone from a random tone to a steady tone (the number 160 is the number of the note we're playing here).

LINE 90 is a little tricky, so follow closely. The first thing we did was use our DIRECTION FORMULA which is shown in the "arrows" diagram above. You can see from the diagram that if you want a symbol to move STRAIGHT RIGHT across the screen, you have to increase the screen location by adding 1. That's what LF=LF+1 means. Of course, to keep the color-location matching you have to increase that by one, also. We also set an END OF LINE LIMIT by using an IF...THEN statement. This statement translates, "If the screen position of the diamond (LF) reaches the end of the line and tries to wrap around to the first location on the next line (E) then GOTO Line 20 and choose a new random location. Note that the diamond stays in the last position on the line because we only erased LF minus 1, not LF. The last position is still LF so the diamond stays there. Is this beginning to resemble a game?

Line 100 sends the program back to Line 70 to POKE the NEW LF (caused by LF=LF+1 in Line 90). There are two GOTOs involved here. GOTO70 sends the program back to move LF one space to the right . . . and GOTO20 in Line 90 sends the program back to get a new random starting point if the diamonds reach the end of the line.

That should be enough to give you something to experiment with, but let's add a few more intriguing ideas before leaving you. You probably noticed that sometimes the little diamonds repeat themselves on lines where there is already a diamond. That's because everything is happening COMPLETELY at random. You can make the diamonds avoid lines which already have diamonds on them by adding this line to your program:

```
95 IFPEEK(E)=90THENGOTO20
```

The PEEK command may be new to you . . . it's explained in the accompanying VIC Magician article in this issue . . . and in this program we are "PEEKing" at the E-1 location (the last position on the line of the current LF diamond) to see if there is already a diamond there. If there is a diamond there . . . in other words, if the PEEK of location E already has a diamond (number 90) in it, then the program is told to go back to Line 20 and start over at another location. If it keeps finding lines with diamonds it will keep looking for a line which doesn't have a diamond on it because everytime it PEEKs at the end of the line it wants to start on and sees a diamond symbol there, it will start again.

Combining Movements

Let's "mix and match" a couple of movements and see what happens. Let's change the diamonds to ARROWS and keep the same LEFT TO RIGHT movement program we've been using. Then let's put BLUE HEARTS all over the screen using the A (ANYWHERE ON THE SCREEN) variable. Finally, let's say that whenever an arrow hits a blue heart, the blue heart turns RED. How in the world do we do all this? Easy. We use our new PEEK command to "look" at the screen location directly in front of the arrow as it's moving. If we PEEK ahead and see a heart, we'll change its color. Sounds simple enough in plain English, doesn't it? Let's see how it works in a program . . .

Continued next issue . . .

For Beginning Gamewriters

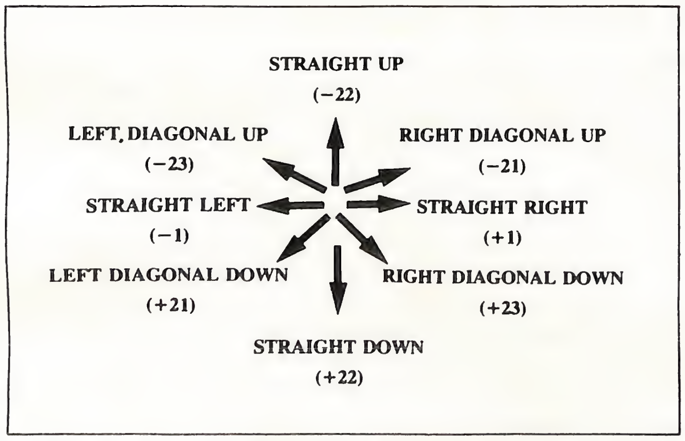
How Computer Animation Works

TWO POKE COMMANDS are required to display or move a symbol on your TV screen. Here's the format you need to use to POKE a heart in the upper right corner:

POKE 7680,83:POKE38400,5

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
7680	7681	7682	7683	7684	7685	7686	7687	7688	7689	7690	7691	7692	7693	7694	7695	7696	7697	7698	7699	7700	7701
7702	7703	7704	7705	7706	7707	7708	7709	7710	7711	7712	7713	7714	7715	7716	7717	7718	7719	7720	7721	7722	7723
7724	7725	7726	7727	7728	7729	7730	7731	7732	7733	7734	7735	7736	7737	7738	7739	7740	7741	7742	7743	7744	7745
7746	7747	7748	7749	7750	7751	7752	7753	7754	7755	7756	7757	7758	7759	7760	7761	7762	7763	7764	7765	7766	7767
7768	7769	7770	7771	7772	7773	7774	7775	7776	7777	7778	7779	7780	7781	7782	7783	7784	7785	7786	7787	7788	7789
7790	7791	7792	7793	7794	7795	7796	7797	7798	7799	7800	7801	7802	7803	7804	7805	7806	7807	7808	7809	7810	7811
7812	7813	7814	7815	7816	7817	7818	7819	7820	7821	7822	7823	7824	7825	7826	7827	7828	7829	7830	7831	7832	7833
7834	7835	7836	7837	7838	7839	7840	7841	7842	7843	7844	7845	7846	7847	7848	7849	7850	7851	7852	7853	7854	7855
7856	7857	7858	7859	7860	7861	7862	7863	7864	7865	7866	7867	7868	7869	7870	7871	7872	7873	7874	7875	7876	7877
7878	7879	7880	7881	7882	7883	7884	7885	7886	7887	7888	7889	7890	7891	7892	7893	7894	7895	7896	7897	7898	7899
7900	7901	7902	7903	7904	7905	7906	7907	7908	7909	7910	7911	7912	7913	7914	7915	7916	7917	7918	7919	7920	7921
7922	7923	7924	7925	7926	7927	7928	7929	7930	7931	7932	7933	7934	7935	7936	7937	7938	7939	7940	7941	7942	7943
7944	7945	7946	7947	7948	7949	7950	7951	7952	7953	7954	7955	7956	7957	7958	7959	7960	7961	7962	7963	7964	7965
7966	7967	7968	7969	7970	7971	7972	7973	7974	7975	7976	7977	7978	7979	7980	7981	7982	7983	7984	7985	7986	7987
7988	7989	7990	7991	7992	7993	7994	7995	7996	7997	7998	7999	8000	8001	8002	8003	8004	8005	8006	8007	8008	8009
8010	8011	8012	8013	8014	8015	8016	8017	8018	8019	8020	8021	8022	8023	8024	8025	8026	8027	8028	8029	8030	8031
8032	8033	8034	8035	8036	8037	8038	8039	8040	8041	8042	8043	8044	8045	8046	8047	8048	8049	8050	8051	8052	8053
8054	8055	8056	8057	8058	8059	8060	8061	8062	8063	8064	8065	8066	8067	8068	8069	8070	8071	8072	8073	8074	8075
8076	8077	8078	8079	8080	8081	8082	8083	8084	8085	8086	8087	8088	8089	8090	8091	8092	8093	8094	8095	8096	8097
8098	8099	8100	8101	8102	8103	8104	8105	8106	8107	8108	8109	8110	8111	8112	8113	8114	8115	8116	8117	8118	8119
8120	8121	8122	8123	8124	8125	8126	8127	8128	8129	8130	8131	8132	8133	8134	8135	8136	8137	8138	8139	8140	8141
8142	8143	8144	8145	8146	8147	8148	8149	8150	8151	8152	8153	8154	8155	8156	8157	8158	8159	8160	8161	8162	8163
8164	8165	8166	8167	8168	8169	8170	8171	8172	8173	8174	8175	8176	8177	8178	8179	8180	8181	8182	8183	8184	8185

Screen Location Chart



Add these values to the Screen location to move symbols in the direction indicated by the arrow.

There are FOUR numbers involved here: the SCREEN LOCATION (7680), POKE VALUE OF THE SYMBOL (83), COLOR LOCATION (38400) and COLOR POKE VALUE (5). Here's how they work:

1. SCREEN LOCATION. There are 506 locations, numbered from 7680 in the top righthand corner to 8185 in the bottom right corner . . . the accompanying chart shows all the location numbers.

2. POKE VALUE OF THE SYMBOL you want to display. See the chart on page 141 of your VIC owner's guide. Example: the POKE value of a solid ball is 81.

3. COLOR LOCATION. This matches the screen location but is easy to calculate because color location is ALWAYS THE SCREEN LOCATION NUMBER ADDED TO THE NUMBER 30720. Example: color location for screen location 7680 is 7680+30720 or 38400.

4. POKE VALUE OF THE COLOR you want to use. Color numbers are: 0-black, 1-white, 2-red, 3-cyan, 4-purple, 5-green, 6-blue and 7-yellow.

To POKE a purple ball in the upper right corner of the screen, type this: POKE7701,81:POKE38421,4 and press the RETURN key (or put these commands in a line of a BASIC program). POKE7701,81 means you are POKEing the number of the ball symbol (81) into screen location 7701 (see chart). POKE38421,4 means you are POKEing the color 4 into the color location that matches the screen location (38421=30720 + 7701).

Moving Symbols

To move a symbol in one of the directions shown by the arrows in the accompanying diagram, simply POKE the symbol on the screen, then add or subtract the values shown for both the SCREEN AND COLOR LOCATIONS. To simulate animation, you then ERASE the symbol from its previous position by POKEing the number 32 (a blank space) into the previous location. Type this:

POKE7910,83:POKE38630,4 (and press RETURN)

To MOVE this symbol one space down and to the right, type this:

**POKE7933,83:POKE38663,4:POKE7910,32
(and press RETURN)**

The first two POKEs added the value 23 to screen and color locations (as indicated by the movement diagram) and the third POKE erased the symbol by POKEing a 32 (blank space) into the previous location. Smoother, more efficient animation is achieved by using FOR . . . NEXT loops to repeatedly add or subtract a location (see examples in this article).

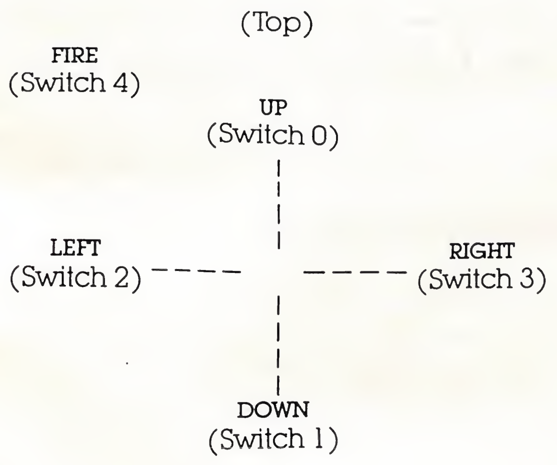
Once again programmer Bill Hindorff came through—this time with an excerpt from the forthcoming Commodore 64 Programmers Reference Guide. *(Yes, all 300-or-so pages of it.) This particular chapter should be of interest to all our game fanatics, who can now go bonkers with those terrific high-res graphics on the 64.*

Joysticks, Paddles, and Light-Pen

The Commodore 64 has two 9 pin Game Ports which allow the use of joysticks, paddles, or a light pen. Each port will accept either one joystick or one paddle. A light pen can be plugged into Port A for special graphic control, etc. . . Examples of using the joysticks and paddles from BASIC and machine language follow

The digital joystick is connected to CIA #1 (MOS 6526 Complex Interface Adapter). This I/O device also handles paddle fire buttons and keyboard scanning. The 6526 has 16 registers which occupy memory locations 56320 thru 56335 inclusive ($DC00 to $DC0F). Port A data appears at location 56320 ($DC00) and Port B data is at location 56321 ($DC01).

A digital joystick is made up of five switches, four for direction and one for the fire button. The joystick switches are arranged as shown:



These switches correspond to the lower 5 bits of the data in location 56320 or 56321. Normally the bit is set to a one if a direction is NOT chosen or the fire button is NOT pressed. When the fire button is pressed, the bit changes to a zero (bit 4 in this case). To read the joystick from BASIC, the following subroutine may be used:

```
10 FORK=0TO10:REM SET UP DIRECTION STRING
20 READ DR$(K):NEXT
30 DATA"","N","S","","W","NW"
40 DATA"SW","","E","NE","SE"
50 PRINT"GOING...":
60 GOSUB 100:REM READ THE JOYSTICK
65 IFDR$(JV)=""THEN80:REM CHECK IF A DIRECTION
WAS CHOSEN
70 PRINTDR$(JV):" ";:REM OUTPUT WHICH DIRECTION
80 IFFR=16THEN60:REM CHECK IF FIRE BUTTON WAS PUSHED
90 PRINT"-----F-----I-----R-----E-----!!!":GOTO60
100 JV=PEEK(56320):REM GET JOYSTICK VALUE
110 FR=JVAND16:REM FORM FIRE BUTTON STATUS
120 JV=15-(JVAND15):REM FORM DIRECTION VALUE
130 RETURN
READY.
```

The values for JV correspond to these directions:

JV EQUAL TO	DIRECTION
0	NONE
1	UP
2	DOWN
3	—
4	LEFT
5	UP & LEFT
6	DOWN & LEFT
7	—
8	RIGHT
9	UP & RIGHT
10	DOWN & RIGHT

A small machine code routine which will accomplish the same task is as follows:

```
1000 .PAGE 'JOYSTICK.8/5'  JOYSTICK - BUTTON READ ROUTINE
1010 ;
1020 ;AUTHOR - BILL HINDORFF
1030 ;
1040 DX=$C110
1050 DY=$C111
1060 *=$C200
1070 DJRR LDX #0      ;THIS ROUTINE READS AND DECODES THE
1080      LDY #0      ;JOYSTICK/FIREBUTTON INPUT DATA IN
1090      LDA $DC00   ;GET READING FROM PORT A
1100      LSR A       ;THE ACCUMULATOR. THIS LEAST
SIGNIFICANT
1110      BCS DJR0    ;5 BITS CONTAIN THE SWITCH CLOSURE
1120      DEY         ;INFORMATION. IF A SWITCH IS CLOSED IT
1130 DJR0 LSR A       ;PRODUCES A ZERO BIT. IF A SWITCH
IS OPEN
1140      BCS DJR1    ;IT PRODUCES A ONE BIT. THE JOYSTICK
DIR-
1150      INY         ;ECTIONS ARE RIGHT, LEFT, FORWARD,
BACKWARD
1160 DJR1 LSR A       ;BIT3=RIGHT, BIT2=LEFT, BIT1=BACKWARD,
1170      BCS DJR2    ;BIT0=FORWARD AND BIT4=FIRE BUTTON.
1180      DEX         ;AT RTS, DX AND DY CONTAIN 2'S
COMPLEMENT
1190 DJR2 LSR A       ;DIRECTION NUMBERS I.E. $FF=-1, $00=0,
$01=1.
1200      BCS DJR3    ;DX=-1 (MOVE RIGHT), DX=1 (MOVE LEFT)
1210      INX         ;DX=0 (NO X CHANGE). DY=-1 (MOVE
UP SCREEN),
1220 DJR3 LSR A       ;DY=1 (MOVE DOWN SCREEN), DY=0
(NO Y CHANGE).
1230      STX DX      ;THE FORWARD JOYSTICK POSITION
CORRESPONDS
1240      STY DY      ;TO MOVE UP THE SCREEN AND THE BACKWARD
1250      RTS         ;POSITION TO MOVE DOWN SCREEN.
1260 ;
1270 ;AT RTS TIME THE CARRY FLAG CONTAINS THE FIRE
BUTTON STATE.
1280 ;IF C=1 THEN BUTTON NOT PRESSED. IF C=0 THEN PRESSED.
1290 ;
1300 .END
READY.
```

Paddles

A paddle is connected to both CIA #1 and the SID chip (MOS 6581 Sound Interface Device) through a game port. The paddle value is read via the SID registers $D419 and $D41A. PADDLES CANNOT BE READ RELIABLY FROM BASIC ALONE!!!! The best way to use paddles, from BASIC or machine code, is to use the following machine language routine. . .(SYS to it from BASIC then PEEK the memory locations used by the subroutine).

```
1000 :*******************************************************
1010 :* FOUR PADDLE READ ROUTINE (CAN ALSO BE USED FOR TWO)
1020 :*******************************************************
1030 :AUTHOR - BILL HINDORFF & JOE MCENERNEY
1040 PORTA=$DC00
1050 CIDDRA=$DC02
1060 SID=$D400
1070 *=$C100
1080 BUFFER *=*+1
1090 PDLX *=*+2
1100 PDLY *=*+2
1110 *=$C000
1120 PDLRD
1130     LDX #1         :FOR FOUR PADDLES OR TWO ANALOG JOYSTICKS
1140 PDLRD0             :ENTRY POINT FOR ONE PAIR (CONDITION X 1ST
1150     SEI
1160     LDA CIDDRA     :GET CURRENT VALUE OF DDR
1170     STA BUFFER     :SAVE IT AWAY
1180     LDA #$C0
1190     STA CIDDRA     :SET PORT A FOR INPUT
1200     LDA #$80
1210 PDLRD1
1220     STA PORTA      :ADDRESS 1ST PAIR OF PADDLES
1230     LDY #$80       :WAIT A WHILE
1240 PDLRD2
1250     NOP
1260     DEY
1270     BPL PDLRD2
1280     LDA SID+25     :GET X VALUE
1290     STA PDLX,X
1300     LDA SID+26     :GET Y VALUE
1310     STA PDLY,X
1320     LDA PORTA      :TIME TO READ PADDLE FIRE BUTTONS
1330     ORA #$80       :MAKE IT THE SAME AS OTHER PAIR
1340     STA PDLY+2     :BIT 2 IS PDL X, BIT 3 IS PDL Y
1350     LDA #$40
1360     DEX            :ALL PAIRS DONE?
1370     BPL PDLRD1     :NO
1380     LDA BUFFER
1390     STA CIDDRA     :RESTORE PREVIOUS VALUE OF DDR
1400     LDA PORTA+1    :FOR 2ND PAIR -
1410     STA PDLY+3     :BIT 2 IS PDL X, BIT 3 IS PDL Y
1420     CLI
1430     RTS
1440 .END
READY.
```

The paddles can be read by using the following BASIC program:

```
10 C=12*4096:REM SET PADDLE ROUTINE START
11 REM POKE IN THE PADDLE READING ROUTINE
15 FORI=0TO63:READA:POKEC+I,A:NEXT
20 SYSC:REM CALL THE PADDLE ROUTINE
30 P1=PEEK(C+257):REM SET PADDLE ONE VALUE
40 P2=PEEK(C+258):REM  "      "   TWO   "
50 P3=PEEK(C+259):REM  "      "  THREE  "
60 P4=PEEK(C+260):REM  "      "   FOUR  "
61 REM READ FIRE BUTTON STATUS
62 S1=PEEK(C+261):S2=PEEK(C+262)
70 PRINTP1,P2,P3,P4:REM PRINT PADDLE VALUES
72 REM PRINT FIRE BUTTON VALUES
75 PRINT:PRINT"fire a ";S1,"fire b ";S2
80 FORW=1TO50:NEXT:REM WAIT A WHILE
90 PRINT"S":PRINT:GOTO 20:REM CLEAR THE SCREEN AND DO AGAIN
95 REM DATA FOR MACHINE CODE ROUTINE
100 DATA162,1,120,173,2,220,141,0,193,169,192,141,2,220,169
110 DATA128,141,0,220,160,128,234,136,16,252,173,25,212,157
120 DATA1,193,173,26,212,157,3,193,173,0,220,9,128,141,5,193
130 DATA169,64,202,16,222,173,0,193,141,2,220,173,1,220,141
140 DATA6,193,88,96
READY.
```

A Little Exercise in Machine Language

Jim Butterfield, Toronto

The first language you meet when you turn on your PET, VIC, or CBM is BASIC. It's powerful and convenient. But deep inside your computer is another language, faster and more powerful, which is working behind the scenes to make the good things happen.

We'll do an experiment to get a glimpse of this inner superlanguage, called "machine language" or more accurately "6502 machine language"

If you have a VIC, remove any memory expansion.

Now: type the following lines:

```
10 REM XXXXXXXXXXXXXXXXXXXXXXXXX
20 SYS 1031
30 PRINT "THAT'S ALL"
```

Line 10 should contain 25 X's—count them to make sure they are right. Now: a special number for line 20: for all PETs, 1031 is the correct number as shown. For VICs, the line should read SYS 4103, and on the Commodore 64 the line should be SYS 2055. Type in the correct value, complete the program, and list it back to be sure it's right.

The number you have typed in line 20 isn't a mystery—it's the address of part of the program itself. In fact, it's the address of the first of the X's that you typed in. We're going to check that to make sure everything is right.

Type in the following direct statement, using the number from line 20:

A=1031 (or 4103, or 2055, as appropriate)

Now we'll check that everything is OK by typing:

PRINT PEEK (A); PEEK (A+23)

The computer should print two numbers, both of which are 88. What does this mean? That's how the letter X is held in the computer's memory . . .we're checking to see that we have the X's in the right place.

Putting in the program.

Now, here's the gimmick: we're going to fit a machine language program right inside a line of BASIC! We'll POKE it into place, byte by byte, and we'll use the value of A that was previously set. Type the following lines carefully:

```
POKE A,32        POKE A+9,16
POKE A+1,228     POKE A+10,32
POKE A+2,255     POKE A+11,210
POKE A+3,201     POKE A+12,255
POKE A+4,13      POKE A+13,202
POKE A+5,208     POKE A+14,208
POKE A+6,1       POKE A+15,250
POKE A+7,96      POKE A+16,240
POKE A+8,162     POKE A+17,238
```

Whew! That's it, but we'd better make one more check to be sure we have put it in OK. Type: T=0:For J=A TO A+17:T=T+PEEK(J):NEXT J:PRINT T and the machine should reply with value 2847, which shows that you didn't make any mistakes.

You may LIST your BASIC program again now. . .but there's an amazing change. Most of the X's have disappeared, and in their place is an astounding hodgepodge of stuff. Don't worry about it—that's where we have been POKE-ing about. It looks like a mess to us, but to BASIC it's just a REM—a remarks line of no particular interest.

Running it.

Type RUN, and the program will start. Nothing will seem to happen, but touch a key and—pow—the key is repeated sixteen times on the screen. Talk about speed typing!

You may terminate the program by pressing the RETURN key—the machine language program is looking for this key and will quit when it sees it. Before you stop things, you might note another interesting thing: machine language ignores the RUN/STOP key unless special arrangements are made to check it.

If by any chance the program does not work, there must be a mistake somewhere. Turn the computer off and back on again. No harm has been done; but you'll have to start all over.

How it works (Part 1).

We placed a machine language program inside a BASIC REM line. BASIC, of course, ignores everything inside a REM line, so it wasn't bothered. . .it proceeded to line 20.

Line 20 contained a SYS command. That's like a subroutine call: it's very much like GOSUB. GOSUB tells you to go to a BASIC subroutine, which will contain a RETURN statement when it's finished. SYS tells you to go to a machine language subroutine, which will give its own kind of return statement (called an RTS instruction).

So we go to the machine language subroutine—tucked inside the REM statement—do whatever it says, and eventually return to BASIC which finishes the job.

How it works (Part 2).

You don't need to know this kind of detail, but for serious students I'll give a detailed rundown on what the machine language program is up to. I'll show the POKE values; then I'll show the same numbers using a special numbering system called "hexadecimal" which machine language programmers use; and finally, a brief explanation of each instruction.

32 228 255	20 E4 FF	Get from keyboard.
201 13	C9 0D	If it's a RETURN key. . .
208 1	D0 01	.skip next line;
96	60	Return to BASIC
162 16	A2 10	Set X count to 16
32 210 255	20 D2 FF	Print the character;
202	CA	Count down X
208 250	D0 FA	If more, back to PRINT
240 238	F0 EE	If not, back to start

It's not as legible as BASIC, but you can see that it has the same kind of clear logical structure that we use in BASIC.

If you can find the location where you placed the 16, you can try higher numbers there (maximum 255). Careful: variable A will have lost its value, so you must reset it or calculate the location yourself. To be sure, PEEK before you POKE.

Summary.

It's quite compact for some jobs: the whole Machine Language program fits inside a line of BASIC.

It's amazingly fast: BASIC couldn't print at that speed.

Perhaps most useful of all: Machine Language can give you a glimpse of the inner workings of your computer.

"Why buy just a video game when you can get a full colour computer for the same price."

A computer like this would have been science fiction a few years ago. Now it's a reality. It's the Commodore VIC-20, a full-fledged, expandable colour computer that costs the same as a video game.

Everybody loves video games and the VIC-20 has some of the best. But the Commodore VIC-20 can also help the kids with their homework and mum with her budgeting. Dad can even take the light, portable VIC-20 to the office for financial and business applications.

And Commodore has many more applications on the way.

With Full Capability For:

- Education programs
- Recreational programs
- Personal computing
- Includes Microsoft, PET BASIC
- Full-size typewriter-style keyboard
- Easy to follow instruction manual
- Memory expansion to 32K RAM
- Connects to any TV set
- 66 graphic characters
- 25K total memory
- 4 sound generators
- 16 colours

VIC 20

A real computer for the price of a toy.

The VIC-20 is the friendliest way we know to learn computing. It has a full computer keyboard even a small child can operate.

It plays music, has exciting graphics and lets you create pictures. It even tells you when you've made a mistake and how to correct it. The VIC-20 can take your children from pre-school through post-graduate studies.

Why get just another game that could end up in the closet? Get an honest-to-goodness home computer for just $299. Get the Commodore VIC-20.

Learn more about Commodore, the home-computer you can depend on. Call or write for the name and location of your Commodore dealer nearest you.

**The Commodore Information Centre,
5 Orion Rd., Lane Cove, NSW. 2066 Tel: 427 4888**

commodore
COMPUTER

MLVL1655

CONCURRENT CHIMER

by Dirk Williams

The following program is a concurrent clock for the unexpanded VIC 20. Not to be confused with workaday boring concurrent clocks – this one plays westminster chimes on the hour.

For those who are unfamiliar with the principle of concurrency (nothing to do with Mr Hawke) it simply means that this program runs even while other programs are running. Thus the clock will be displayed on the top line of the VICs screen and will chime hourly even while you are typing in another program or working out your budget or playing blue meanies.

In order to achieve this, the program uses the interrupt routine (this is a routine which is performed transparently every 60th of a second by the VIC) which then drives the concurrent clock. The clock obtains its accuracy from the fact that it occurs so regularly.

The two listings contain the said program – the basic listing should be used by people without machine code monitors. When run, the basic program pokes the clock program into high memory space – and sets the top of memory and array pointers so that the code will not be overwritten. The poke 788,157 and POKE 789,28 re–vector the interrupt routine through the clock program. The listing has been included to allow advanced programmers to change, if necessary, the program.

The colour of the clock on the top line may be changed by poking a number (default 0) into 7472. RUN STOP/RESTORE will turn the clock off – so simply poke 788,157:POKE789,28 to turn it back on. The clock will slow down during disk access – but this is to be expected.

```
1DA8 37 00 24 00 11      ; IDA8=jiffies  IDAA secs  IDAC mins  IDAE hours
1DAD 00 02 10 B6 B3      ; 1DB0=ASCII secs(units)  1DB1=ASCII secs(tens)
1DB2 B7 B1 B2 B0 00      ;etc
1DB7 00 40 EC EC 80
1DBC 40 EC CF FF 40      ;1DB6=chime flag  1DB7=counter for notes
1DC1 A9 D4 00 80 FF      ;1DB8=data table(sets of 4 bytes) as below
1DC6 FF 00 20 00 00      ;length, voice 1, voice 2, voice 3, etc
1DCB 00 40 FF FF 00
1DD0 40 A9 D4 00 40
1DD5 EC EC 80 40 EC
1DDA CF FF 10 00 00
```

```
1C9D PHP
1C9E PHA
1C9F TYA
1CA0 PHA
1CA1 TXA
1CA2 PHA
1CA3 INC $1DA8          ;increment jiffies
1CA6 LDA $1DA8
1CA9 CMP #$3C           ;compare with 60
1CAB BNE $1CE4          ;jump to write routines if less
1CAD LDA #$00           ;zero jiffies
1CAF STA $1DA8
1CB2 INC $1DAA          ;increment seconds
1CB5 LDA $1DAA
1CB8 CMP #$3C           ;compare with 60
1CBA BNE $1CE4          ;jump if less
1CBC LDA #$00
1CBE STA #$1DAA         ;zero seconds
1CC1 INC $1DAC          ;increment minuites
1CC4 LDA $1DAC
1CC7 CMP #$3C           ;compare with 60
1CC9 BNE $1CE4          ;jump if less
1CCB LDA $$00           ;zero minuites
1CCD STA $1DAC
1CD0 INC $1DAE          ;increment hours
1CD3 LDA #$01           ;set chime flag
1CD5 STA #$1DB6
1CD8 LDA #$1DAE
1CDB CMP #$0D           ;compare with 13
1CDD BNE $1CE4          ;jump if less
1CDF LDA #$01           ;set to one
1CE1 STA $1DAE
1CE4 JSR $1D3C          ;jump to ASCII convert routine
1CE7 JSR $1CF6          ;jump to screen write routine
1CEA JSR $1D66          ; jump to chime routine
1CED PLA
1CEE TAX
1CEF PLA
1CF0 TAY
1CF1 PLA
1CF2 PLP
1CF3 JMP $EABF          ;return to normal interrupt - make this JMP $6346
                        if entered from MLM
1CF6 LDX #$00           ;this routine writes to screen
1CF8 LDA #$A0           ;clear in front of clock with RVS spaces
1CFA STA $1E00,X
1CFD INX
1CFE CPX #$07
1D00 BNE $1CFA
1D02 LDX #$05
1D04 LDY #$07
1D06 LDA $1DB0,X
1D09 STA $1E00,Y        ;put clock on screen
1D0C DEX
1D0D INY
```

```
1D0E LDA $1DB0,X
1D11 STA $1E00,Y
1D14 CPX #$00
1D16 BEQ $1D23
1D18 DEX
1D19 INY
1D1A LDA #$BA
1D1C STA $1E00,Y
1D1F INY
1D20 JMP $1D06
1D23 LDA #$A0                ;clear behind clock
1D25 LDX #$0F
1D27 STA $1E00,X
1D2A INX
1D2B CPX #$16
1D2D BNE $1D27
1D2F LDA #$00                ;load colour into A
1D31 LDX #$00
1D33 STA $9600,X             ;set up colour RAM
1D36 INX
1D37 CPX #$16
1D39 BNE $1D33
1D3B RTS
1D3C LDX #$00                ;this routine does ASCII conversion on time bytes
                             ;and stores results from 1DB0 onwards
1D3E LDA $1DAA,X
1D41 LDY #$00
1D43 CMP #$0A
1D45 BMI $1D52
1D47 CMP #$00
1D49 BEQ $1D52
1D4B CLC
1D4C SBC #$09
1D4E INY
1D4F JMP $1D43
1D52 CLC
1D53 ADC #$B0
1D55 STA $1DB0,X
1D58 TYA
1D59 CLC
1D5A ADC #$B0
1D5C STA $1DB1,X
1D5F INX
1D60 INX
1D61 CPX #06
1D63 BNE $1D3E
1D65 RTS
1D66 LDX $1DB6               ;this routine is the chime routine
1D69 CPX #$00                ;check chime counter/byte check flag
1D6B BEQ $1DA7               ;jump if not set
1D6D LDY $1DB7,X             ;get note length from table start $1DB8
1D70 LDA $1DB8,X             ;get voice 1 note
1D73 STA $900A               ;store it, etc
1D76 LDA $1DB9,X
1D79 STA $900B
```

```
1D7C LDA $1DBA,X
1D7F STA $900C
1D82 LDA #$0F
1D84 STA $900E             ;set volume
1D87 INC $1DB7
1D8A CPY $1DB7             ;check if note finished
1D8D BEQ $1D90
1D8F RTS
1D90 LDA #$00              ;zero counte if note finished
1D92 STA $1DB7
1D95 CLC
1D96 LDA $1DB6
1D99 ADC #$04              ;increment check flag to point to next note
1D9B STA $1DB6
1D9E CMP #$28              ;song beneath flag - could be changed for diff
                           ;tune
1DA0 BMI $1DA7
1DA2 LDA #$00
1DA4 STA $1DB6             ;zero check flag
1DA7 RTS
```

(Cont. on page 60)

OMEGA RACE The Finer Points

by David Berezowski
Present Omega Race Champ

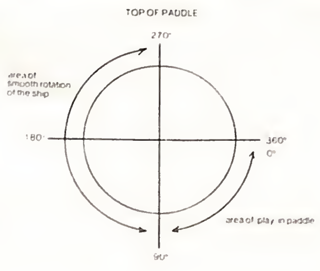
Tired of getting wiped out by the DROID FORCE? Want to hit 40,000 and get that much-deserved free ship? Or better yet, want to get six-digit scores? Here's how to do it! (Or at least how I did it!)

1. Get rid of that joystick! Grab firm hold of the game paddle, it's the OMEGA WARRIOR'S controller!

Here are some points to remember when using the paddle.

P1. The paddle doesn't rotate 360 degrees. It has what one might call a limited area of movement. It is this 'limited area of movement' that the young OMEGAN must learn to control and later master to become a WARRIOR!

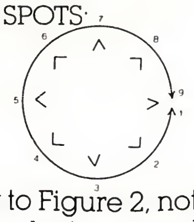
FIGURE 1



Referring to Figure 1, you can see that the paddle only rotates 270 degrees. Of this 270 degrees, approximately 180 degrees gives you a smooth rotation of your ship. The other 90 degrees is sort of like the play in a car's steering wheel. Don't be fooled by the play in the paddle.

P2. DEAD SPOTS·

FIGURE 2



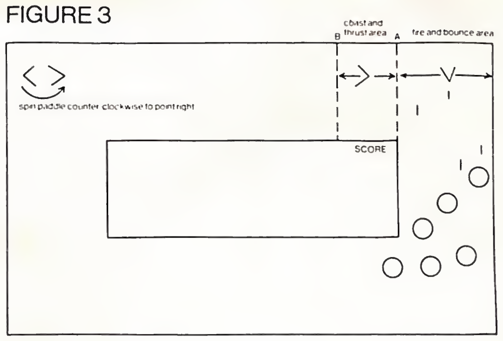
Referring to Figure 2, note that as you turn the paddle clockwise, your ship turns clockwise and vice-versa. HOWEVER, if you have turned your ship clockwise to position 9, and want to continue on to position 2, you can't! Note that the same is true if you're at 1 and want to go to 8. Therefore, when pointing left (<) remember what position you're at (either 1 or 9). If you're at 1 and want to go to 8, you'll have to quickly spin the paddle clockwise, and vice-versa if you're at 9 and want to go to 2.

2. When starting a new screen or round, always point your ship left (<). If the round begins with your ship on the right side of the screen

then you are pointing in the right direction. (Proceed to step 3, The Technique). However, if you are on the left side of the screen, then quickly spin your paddle counter-clockwise so that you are pointing right (>). You are now ready for The Technique!

3. The TECHNIQUE: (This is what you've all been waiting for!). The whole secret to what I'm about to say lies in the fact that the DROIDS base their missile firing direction on where you are on the screen at the time they decide to fire at you. If you can get them to fire where they can't hit you, then you can blow them to bits. Here's how to do it!

FIGURE 3



T1. Referring to Figure 3.

a) Make sure you are pointing right (>). (You'll have to quickly spin the paddle counter-clockwise if you were following my instructions earlier!)

b) Thrust to a moderate speed and point your ship down (V), *after you have finished thrusting.*

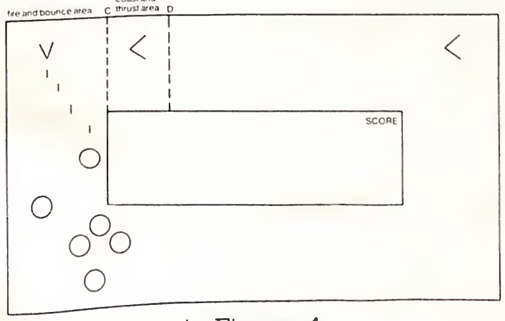
c) When you reach point A, rapidly fire down at the DROIDS. Continue to fire until you have bounced back to point A.

d) Point the ship right (>) (make sure you turn the paddle clockwise to get to this position). When you reach point B, thrust to a moderate speed and point the ship down (V) again (after you have finished thrusting of course).

e) Go to step C until all the DROIDS are dead.

f) Clean up as many 'mines' as you can and prepare for the next round by pointing your ship left (<).

FIGURE 4



T2. Referring to Figure 4.

Use the same technique as T1, except use points C and D instead of A and B, and point your ship left, instead of right. (You shouldn't have to turn your paddle if you were following my instructions earlier!)

Why does the above work??? If you time it right, by bouncing back and forth at a moderate speed, the DROIDS will fire at you while you are right of point A or left of point C. Fortunately they will fire up at an angle and their missiles will harmlessly hit the inner boundary. Thus there will be few (if none at all) missiles threatening you and you are free to fire down at the DROIDS. It's almost like shooting fish in a barrel!

NOTES

N1. Don't fire repeatedly into an explosion. They tend to act like black holes and 'eat' oncoming missiles. Better to space your missiles apart and wait until the explosion has disappeared.

N2. Don't fire more than four times in a row. The system tends to store up the fifth button press. After four missiles have been fired and the first missile 'dies', a fifth missile is mysteriously fired from your ship. THIS CAN REALLY THROW YOUR TIMING OFF! To see this in motion, try the following. Start the game and point your ship so that the missiles will fire along the top of the screen. Now quickly press the fire button five times. Four missiles will be fired and travel across the screen.

As the first missile 'dies', a fifth missile will fire from your ship even though you haven't pressed the fire button!

N3. Pressing shifted F3 (i.e., F4) will give you five ships to start with instead of three. However, when reporting high scores, remember to note how many ships you started with!

N4. Using different color combinations might improve your game. I like a black background with cyan characters.

N5. Try and destroy the flashing DROIDS first. These guys soon turn into DEATH STARS which fly faster then you do and are very deadly. Never fire at a DEATH STAR head on. You must learn to anticipate where he is going and fire at where you think he will be, not where he actually is. This is due to the fact that the DEATH STAR flies so fast, that if you shoot right at him, by the time the missile gets there, he will be somewhere else.

N6. When in real danger, don't sit there like a dummy and shoot. Its much better to run away (firing as you go of course)!

Good Luck
[illegible]

(Cont. from page 58)

```
 5 PRINT"♥       SETTING UP"
 10 POKE49,128:POKE50,28:POKE55,128:POKE56,28:CLR:RESTORE
 20 FORX=7325TO7646
 30 READV:POKEX,V:NEXT
 40 POKE788,157:POKE789,28
 50 PRINT"♥QQQQQQ":INPUT"TIME(HHMMSS)";T$
 60 IFLEN(T$)<>6THEN50
 70 T1=VAL(RIGHT$(T$,2)):IFT1>59THEN50
 75 T2=VAL(MID$(T$,3,2)):IFT2>59THEN50
 80 T3=VAL(LEFT$(T$,2)):IFT3>12THEN50
 85 POKE7594,T1:POKE7596,T2:POKE7598,T3
 90 PRINT"♥":END
 100 DATA8,72,152,72,138,72,238
 110 DATA168,29,173,168,29,201
 120 DATA60,208,55,169,0,141,168
 130 DATA29,238,170,29,173,170
 140 DATA29,201,60,208,40,169,0
 150 DATA141,170,29,238,172,29
 160 DATA173,172,29,201,60,208
 170 DATA25,169,0,141,172,29,238
 180 DATA174,29,169,1,141,182,29
 190 DATA173,174,29,201,13,208
 200 DATA5,169,1,141,174,29,32
 210 DATA60,29,32,246,28,32,102
 220 DATA 29,104,170,104,168,104
 230 DATA40,76,191,234,162,0,169
 240 DATA160,157,0,30,232,224,7
 250 DATA208,248,162,5,160,7
 260 DATA189,176,29,153,0,30,202
 270 DATA200,189,176,29,153,0
 280 DATA30,224,0,240,11,202,200
 290 DATA169,186,153,0,30,200,76
 300 DATA6,29,169,160,162,15
 310 DATA157,0,30,232,224,22
 320 DATA208,248,169,0,162,0
 330 DATA157,0,150,232,224,22
 340 DATA208,248,96,162,0,189
 350 DATA170,29,160,0,201,10
 360 DATA48,11,201,0,240,7,24
 370 DATA233,9,200,76,67,29,24
 380 DATA105,176,157,176,29,152
 390 DATA24,105,176,157,177,29
 400 DATA232,232,224,6,208,217
 410 DATA96,174,182,29,224,0
 420 DATA240,58,188,183,29,189
 430 DATA184,29,141,10,144,189
 440 DATA185,29,141,11,144,189
 450 DATA186,29,141,12,144,169
 460 DATA15,141,14,144,238,183
 470 DATA29,204,183,29,240,1
 480 DATA96,169,0,141,183,29
 490 DATA24,173,182,29,105,4
 500 DATA141,182,29,201,40,48
 510 DATA5,169,0,141,182,29,96
 520 DATA17,0,26,0,1,0,1,16
 530 DATA182,178,177,176,177,176
 540 DATA0,0,64,236,236,128,64
 550 DATA236,207,255,64,169,212
 560 DATA0,128,255,255,0,32,0
 570 DATA0,0,64,255,255,0,64
 580 DATA169,212,0,64,236,236,128
 590 DATA64,236,207,255,16,0,0,0

READY.
```

Behind the Programs

An Interview with Rick Madge, Creator of Garden Wars

Illustration by Robert Hunchar.

Rick Madge

Rick Madge, the 25 year-old creator of Commodore's Garden Wars cartridge game, talks very softly on the phone (our method of communication for this interview), and seems to enjoy playing around on his computer more than he enjoys being interviewed. Nevertheless, he provides some interesting insights into how he came up with this unusual and challenging maze game, and (for those of you who aspire to seeing your name in our High Score column) some tips on how to score better.

Power/Play: For starters, who are you and what do you do in your real life?

Rick Madge: I'm a research engineer—an electrical engineer—for Ontario Hydro. My job right now is to investigate the effects of power lines on radio transmissions.

P/P: With that kind of background, how did you get into writing games?

RM: About three years ago, when I first started with Ontario Hydro, I was using one of their PETs as part of my job. I started playing around with it and decided I wanted one for myself. Before that my only experience with computers had been with the mainframes at university, which are *not* user friendly. The PET was a welcome change.

My first game was a golf game, but it was pretty crude because I was just learning BASIC—I'd only had a brief exposure to FORTRAN at university. Since then I've written a whole mess of games, but they're mainly just for myself, for my own amusement.

I especially like designing adventure-style games, where you program a little maze and wander around, getting attacked by monsters. I've also put together a program for playing the card game Euchre, but that's a 32K game for the PET.

P/P: In the game market the big thrust seems to be toward "space" or "cosmic" kinds of themes. How did you come up with the idea for something as down-to-earth as Garden Wars?

RM: I wrote a program called Rat Man, first. Garden Wars evolved out of that idea. It seemed to me to be something people could identify with, something familiar. And it's hard to come up with a new concept in a space game. Even if it's different, it would *look* like what's already out there.

P/P: How did you decide what elements would go in? For instance, the wiggling snakes impressed me.

RM: The snakes were easy. The truth is, things that seem most impressive are usually easy to do. But things that seem simple—like keeping the program running—are a lot harder.

When I was deciding what would go in, as usual there had to be chasers and chasees. Then, one way of fighting the chasers is to have arrows, so I gave the chasee arrows. But I wanted to have something that wasn't just killing, so I put in paralyzing, instead.

When the bombs explode, they send paralyzing shock waves in all four cardinal directions. Also, when the spiders turn blue, you can't kill them anymore. You can only stun them. And once they turn black, you can't get rid of them at all. Then you just have to run away from them.

I went through a number of things people typically associate with gardens to decide what to put in. I would have liked to put in flying insects that wouldn't be restricted to the boundaries of the maze, but it would have taken too much memory. As it was, I ended up with only about 8 bytes left once it was finished.

P/P: What do you consider the outstanding features of the game?

RM: First, you have all the various creatures that leave behind bombs and eggs. Then the spiders come from the eggs. If you don't kill them, they turn blue or black, and there are more and more of them. Another feature is the eight levels of play, with eight different mazes. I also like the treasures. There are eight of them, also. They're those flashing things you see appearing and disappearing.

P/P: What are the clues to scoring well in all that craziness?

RM: The best attack is to kill creatures and eat eggs. Also, don't shoot treasures, because when you do that they disappear and come up somewhere else. But if you run over a treasure, you get 100 points per treasure per level, and if you get all eight on one level you get 1000 points added at the end of that level. I tried to make it so a person couldn't just spend their whole time shooting. The clue to really racking up points is to get the treasures.

P/P: What do you see as the weaknesses of the game—the things you would have liked to improve?

RM: I would have liked to have had the flying insects come across the screen, but, as I said, that would have taken another 300 bytes that just wasn't there.

P/P: Did you run into any other obstacles as you were working on the game?

RM: None in particular. The main problems were in debugging, because it was my first attempt at programming in assembly language and I was always making dumb mistakes—typos mainly. Occasionally it would take me a week or so to find the mistake, so that held things up sometimes.

P/P: Among the computer games you've played, which one is your favorite?

RM: I usually don't play computer games. I've been too busy programming, and I like to program more than play.

P/P: For all our hot games people, what's your high score on Garden Wars?

RM: So far I've scored 81,000.

Editor's Note: *We've excluded game creators from our High Score competition.*

High Scores

VIC AVENGER
9,060
Bram Koster, Otterville, Ontario

JUPITER LANDER
207,400
Christopher Champlain, St. Petersburg, FL

SUPER ALIEN
45,700
Robert Schaeffer, Brookline, MA

MIDNIGHT DRIVE
14.11 km
Nathan Mehl, Newark, DE

RADAR RAT RACE
122,240
John Higginson, South Holland, IL

SUPER SLOT
7,306 coins
Jerry Krueger, Cary, IL

PINBALL
1,500,000
Joe Ferrari, Commodore, Toronto

MOLE ATTACK
309
Barbara Brey, Phoenix, AZ

DRAW POKER
12,819
Angie Traina, Jonesboro, LA

CAR CHASE
75,865
Zach Coleman, Charlotte, NC

SLITHER
325
Kelly Stanley, Florissant, MO

SUPER SLITHER
129
Robert Schaeffer, Brookline, MA

BLUE MEANIES
800
Jon Alderman, Willowdale, Ontario

GORF
55,000
Joe Ferrari, Commodore, Toronto

OMEGA RACE
3 ships: 194,050
5 ships: 204,980
David Berezowski, Commodore, Toronto

GARDEN WARS
68,430
Joe Ferrari, Commodore, Toronto

We had several high scores come in too late to make this issue. We'll get them into the March issue. If your score didn't set a record this time, keep playing! Maybe you'll topple these champion gamesters next time!

VIC Baseball

by Mark Biggs

Ah, baseball! Remember the sweet spring air, the roar of the crowd? VIC Baseball is a great reminder of the season to come. A few clues: if you pick Team #1 you're the home team, which means the crowd cheers when you get a hit. That means, however, that Team #2 has to put up with cheers when they goof, because the home crowd is very rude toward visitors. The game offers you a choice of pitches and is pretty realistic in responding to your batting abilities. Use the keyboard to pitch and hit: F throws a fast ball, C a change-up, E a curve to the left and R a curve to the right. The letter P swings the bat.

Illustration by Larry Motkowski

```
5 T1=1:VV=1:INPUT"NAME OF TEAM 1";C$
7 INPUT"NAME OF TEAM 2";D$
9 INPUT"#OF INNINGS";TN
10 PRINT"■";:POKE36879,205:POKE36878,15
15 FORX=0TO504:PRINT"■ ";:NEXT:PRINT"■■";
20 W=8151:D=1:Y=0:GOSUB25:GOTO35
25 POKEW+22*Y,32:POKEW+30720+22*Y,1:POKEW+
(2*D)+22*Y,32
27 POKEW+D+30720+22*Y,1:IFD=1THENPOKEW+D+22*Y,
78:GOTO31
29 POKEW+D+22*Y,77
31 W=W+D:Y=Y-1:IFY=-9THENY=0:RETURN
33 GOTO25
35 W=7985:D=-1:GOSUB25:W=7967:D=1:GOSUB25:
W=8153:D=-1:GOSUB25
40 IFVV=TN+1ANDVS=HCTHENPRINT"■■■■■■EXTRA
INNINGS!!■":FORT=1TO2000:NEXT:TN=TN +1:GOTO10
42 IFVV=TN+1THENPRINT"■■■■■■■GAME OVER!■":
IFHC>VSTHENGOSUB420:END
45 POKE7984,90:POKE7968,90:POKE7800,90:FORY=-
1TO9:POKE7976+22*Y,32:POKE38696+22* Y,1:NEXT
46 POKE8085,95:POKE8087,105
47 POKE7976,90:POKE8152,90:POKE8151,81:
POKE8173,66:POKE8175,32
49 POKE38704,1:POKE38520,1:POKE38688,1
50 IFT2=1THENZZ=6:GOTO55
52 ZZ=2
55 POKE38871,ZZ:POKE38893,ZZ:POKE38872,ZZ:
IFFB=1THENPOKE38704,ZZ
57 IFSB=1THENPOKE38520,ZZ
59 IFTB=1THENPOKE38688,ZZ
60 PRINT"OUTS="O,"INNING#"VV,"STR.="SR,"BLS.="BB,
62 IFT2=1THENPRINT"■"C$"="HC,"■■"D$"="VS"■■":
GOTO65
64 PRINT"■■"C$"="HC,"■■"D$"="VS"■
65 GETA$:IFA$=""THEN65
67 FORT=1TOINT(RND(1)*1000)+10:NEXT:X=0:Y=1
70 IFA$="F"THENQQ=1:Q=1:GOTO80
71 IFA$="C"THENQQ=1:Q=35:GOTO80
73 IFA$="R"THENX=1:QQ=5:Q=10:GOTO80
74 IFA$="E"THENX=-1:QQ=5:Q=1:GOTO80
75 GOTO65
80 POKE7976+22*Y,46:FORT=1TOQQ:NEXT:POKE7976+
22*Y,32:Y=Y+1:IFY=5THEN85
82 GETB$:IFB$<>"P"THEN80
84 SR=SR+1:GOTO400
85 POKE7976+22*Y+X,46:FORT=1TOQ:NEXT:
POKE7976+22*Y+X,32:Y=Y+1
87 IFY=10THENT=RND(1):IFT<=.6THENSR=SR+1:
GOTO400
89 IFY=10THENBB=BB+1:GOTO400
90 GETB$:IFB$<>"P"THEN85
91 POKE8085,95:POKE8087,105
95 POKE8173,32:POKE36876,230:FORT=1TO400:
NEXT:POKE36876,0:X=0:B=0:Z=RND(1)
98 IFY=7THENDX=-1:POKE8152,78:GOSUB105:GOTO140
100 IFY=9THENDX=1:POKE8152,77:GOSUB105:GOTO140
101 IFZ<=.15THENSR=SR+1:GOTO300
102 IFY=8THENDX=0:POKE8152,67:GOSUB105:GOTO160
103 SR=SR+1:GOTO300
105 FF=154
106 POKE8130+X+22*B,46:POKE36875,FF:FORT=1TO10:
NEXT:POKE36875,0:FORT=1TO10:NEXT:FF=FF+7
109 POKE8130+X+22*B,32:B=B-1:X=X+DX
110 IFB=-9THENRETURN
115 GOTO106
140 IFZ<=.75THENO=O+1:SR=0:BB=0:GOTO405
143 IFZ<=.85THENSR=SR+1:PRINT"■■■■■■■■■■■■";:
IFSR=3THENSR=2
145 IFZ<.85THENPRINT"FOULBALL■■■■■■■■■";:
FORT=1TO2000:NEXT:PRINT"■          ■■";:GIOTO300
146 IFT1=1THENPOKE36877,240
147 HM=1:SR=0:BB=0:IFZ<=.95THENW=1:GOTO410
149 W=2:GOTO410
160 SR=0:BB=0:IFZ<=.55THENO=O+1:GOTO405
161 IFT1=1THENPOKE36877,240
163 HM=1:IFZ<=.75THENW=1:GOTO410
165 IFZ<=.87THENW=2:GOTO410
167 IFZ<=.95THENW=3:GOTO410
169 W=4:FORL=130TO254:POKE36876,L:FORM=1TO40:
NEXTM:NEXTL:POKE36876,0!200 FF=195
201 FORC=1TOW:X=1:Y=-1
202 CH=1:CF=1:CS=1:CT=1
205 A=1:F=77:IFFB=1THENF=81:CF=ZZ
206 S=78:IFSB=1THENS=81:CS=ZZ
207 TH=77:IFTB=1THENTH=81:CT=ZZ
208 H=78:IFHM=1THENH=81:CH=ZZ
210 POKE8152+X+22*Y,H:POKE7984+Y+22*Y,F:
POKE7800+Y+22*X,S:POKE7968+X+22*X,TH
211 POKE38872+X+22*Y,CH:POKE38704+Y+22*Y,CF:
POKE38520+Y+22*X,CS:POKE38688+X+22*X,CT
212 FORT=1TO5:NEXTT:FORT=1TO10:POKE36875,FF:
NEXTT:POKE36875,0:FF=FF+1
214 IFA=1THENF=77:S=78:TH=77:H=78:CH=1:CF=1:
CS=1:CT=1:A=0:GOTO210
216 X=X+1:Y=Y-1:IFX=8THEN220
218 GOTO205
220 IFTB<>1THEN223
221 POKE36876,250:FORT=1TO500:NEXTT:POKE36876,0
222 IFTB=1ANDT1=1THENGOSUB420
223 SC=SC+TB:TB=SB:SB=FB:FB=HM:HM=0:NEXTC
300 IFO=3ANDT1=1THENT1=0:T2=1:O=0:FB=0:SB=0:
TB=0:PRINT"■";:GOTO40
305 IFO=3ANDT2=1THENT2=0:T1=1:O=0:FB=0:SB=0:
TB=0:PRINT"■";:VV=VV+1:GOTO40
310 IFT1=1THENHC=HC+SC:SC=0
315 IFT2=1THENVS=VS+SC:SC=0
320 IFSR=3THENO=O+1:SR=0:BB=0:GOTO405
325 IFBB=4THENHM=1:W=1:BB=0:SR=0:GOTO410
327 PRINT"■";:GOTO40
400 POKE36876,230:FORT=1TO100:NEXT:
POKE36876,0:GOTO300
405 POKE36874,180:FORT=1TO1000:NEXT:POKE36874,0
406 IFT2=1THENGOSUB420
407 GOTO300
410 FORT=1TOW:POKE36876,235
412 FORX=1TO700:NEXTX:POKE36876,0:FORX=1TO400:
NEXTX:NEXTT
413 IFT2=1THEN415
414 GOSUB420
415 POKE36877,0:POKE8151,32:POKE8152,90:GOTO200
420 POKE36877,240:FORX=1TO1000:NEXT:FORD=
15TO0STEP-1:POKE36878,D:FORT=1TO120:NEX
TT:NEXTD
422 POKE36877,0:POKE36878,15:RETURN

READY.
```

We're Glad You Asked...

Q How does one "trap" the RUN/STOP key on the VIC 20? I am presently working on a preschool program and am using GET statements instead of INPUT statements when I want an operator's response. Unfortunately, though, a child may inadvertently break the program by hitting the RUN/STOP key. I was unable to "trap" the RUN/STOP key in the usual manner, using GET A$ and the CHR$ ("X") function. I suspect one has to POKE the VIC's memory to trap the key, but where?
A You're entirely correct. To disable, POKE 788,194. To re-enable, POKE 788,191. You can use these commands wherever appropriate in your program. Side effect: this also traps the real time clock (TI and TI$).

Q How does one protect a VIC 20 program from being copyable?
A It's virtually impossible to protect any program from a determined thief. This is particularly true with programs on cassette. However, if you want to protect a program from a relatively inexperienced copycat, a simple technique is to put part of it (maybe some subroutines or calculations) into machine language. Because machine language is stored in a different part of memory than BASIC, if someone tries to copy the program by simply SAVEing from tape or disk, only the BASIC part of the program will be saved. And the thief will be foiled because the program will not run without the hidden machine language.

Q I want to use my VIC 20 with expanded memory AND the VICMODEM. How can I override the current prohibition? My work involves preparing texts at home, then calling a university computer to download the texts to add to other files and print on the high-quality printers.
A A fairly simple task. Plug in the memory expander you intend to use with the VICMODEM, type in the Terminal Software program in the VICMODEM manual and save it (with the memory expander in place). When you load and run, you've got the use of both expander and modem.

Q Why are these calculations not exact:

PRINT exponents of
3↑4 = 81.0000001
3↑6 = 729.000001
etc.

A The problem is caused by rounding errors introduced when the computer converts decimal numbers to and from binary numbers. There are several ways around these inaccuracies, but the easiest one is to have the computer print only the first four or five (or seven, or whatever, depending on where the inaccuracy begins) numbers it sees.

Q Is there some way to look at what's in my VIC's memory?
A The best way is to use the VICMON cartridge. But if you don't have one, this little program will permit you to look inside your VIC's memory:

```
10 X=X+1
20 PRINT CHR$(PEEK(X));
30 Y=Y+1:IFY=500THENPRINT
   "CLR/HOME"X:Y=0
40 GOTO 10
```

When you type this in and RUN, the VIC will print on the screen each character stored in each of the 65535 memory locations. Since many locations are not in the VIC itself but in plug-in cartridges, when they are being addressed there will be no display. Also, if you'd like a less flashy but more informative display, change line 20 to:

```
Z=PEEK(X):PRINTZ;
```

This will give you the value in each location, rather than the character itself. For more information on what you're seeing on the screen, check your *VIC 20 Programmer's Reference Guide*, Appendix F. (Thanks to the VIC-NIC News, Box 981, Salem, NH 03079 for part of this tip.)

Q In the Fall 1982 issue of *Power/Play* you said it is not possible to restore the VIC back to its unexpanded state without removing the expander cartridges. Actually, restoring the VIC back to its unexpanded state is possible without removing the expander cartridge. UMI's BASIC Utility Program, BUTI (pronounced "beauty"), includes a command for setting the VIC to its unexpanded, VIC plus 3K, and VIC plus 8K or more configurations (if memory is available) without removing any cartridge.

Also, our experience has indicated that, when an 8K or 16K memory expander is added, the start of screen memory is at 4096 decimal, while the start of BASIC is at 4608 decimal. Your column said that under these conditions the start of BASIC is at 4096.

Sincerely,
David Lundberg
Technical Director,
United Microware Industries, Inc.

A Sorry we missed that simple error concerning the start of BASIC. Thanks for catching it.

AVAILABLE NOW FROM COMPUTER BUSINESS AIDS

WORD PRO – the tried and tested Wordpro series for the C64	**$215.00**
VISICALC and DATA MANAGER – electronic spreadsheet and data base for the C64. Visicalc	**$148.00**
Data Manager	**$236.00**
COCO – computer tutor for the C64	**$123.00**
SPACE BELT – a space game for the C64	**$55.00**
RABBIT – speed cassette loader and saver for VIC20 and C64	**$75.00**

VICTREE – from SKYLES ELECTRICS, so you know it's good.
- **Leaves** your VIC20 or C64 with 42 additional commands;
- **Branches** out to most Basic 4.0 programs;
- **Roots** into most printers.

PET SPEED COMPILER – for the C64	**$353.00**
INTERPOD – a connector for the C64 or VIC20 to provide IEEE/RS232C AND IEEE serial I/O	**$340.00**
VIE (VIC20 IEEE) and CIE (CBM 64 IEEE) – gives your computer IEEE–488 I/O for all Commodore and IEEE applications	**$172.50**
THE MINI JINI – plug in cartridge for VIC20 and CBM64. A data base which is very user friendly, with established data base for ham radio, etc. Can save data on cassette or disk.	**$150.00**
BLANK CARTRIDGES FOR THE VIC20. 4K empty or 16K boards to allow you to put programs on cartridges. **SOFTWARE HOUSES** – we can also put your **BASIC** programs onto cartridges with auto start. 4K empty cartridges for VIC20	**$16.00**
8K cartridges	**$63.00**

SOON AVAILABLE – COMPLETE DISASSEMBLY OF C64

Direct enquiries to :

COMPUTER BUSINESS AIDS
Suite 11, Ocean Plaza,
6th Avenue,
MAROOCHYDORE QLD 4558

Phone: (071) 435 849

(Dealers Note): All retail sales except in Sunshine Coast area will be forwarded to authorized Commodore dealers.

ONE STOP SERVICE

Never before in the history of computers has anyone been able to offer users so much.

Computer Reporting Services
has collected the largest range of VIC–20 and COMMODORE 64 books in Australia.

From the introductory to the intricate; now in one office a complete collection of literature for Commodore Computers.

Vic Innovative Computing	*($17.95)*
Vic Innovative Cassette 1	*($20.00)*
Vic Innovative Cassette 2	*($20.00)*
Vic Innovative Cassette 3	*($20.00)*
The Wizard and the Princess Cassette	*($20.00)*
The Games Pack Cassette	*($20.00)*
Learn Computer Programming with the Commodore VIC	*($5.95)*
VIC–20 Programmers Reference Guide	*($22.00)*
Vic Revealed (Available June)	*($18.95)*
Vic Graphics (Available July)	*($18.95)*
Getting aquainted with your VIC–20	*($19.95)*
Zap Pow Boom (30 Great Games)	*($24.95)*
50 Outstanding Games for the VIC–20	*($19.95)*
VIC–20 Users Guide	*($23.20)*
Commodore 64 Programmers Reference Guide	*($22.00)*
Commodore 64 Users Guide	*($19.95)*
Commodore 64 Games Book	*($19.95)*

Retail prices are correct at time of publication.

Please visit our showroom
Suite 204, 284 Victoria Avenue,
Chatswood NSW 2067
or call us (02) 419 5966

See back cover for more details and order form